

Effizienter Einsatz von Optimierungsmethoden in der Produktentwicklung durch dynamische Parallelisierung

Dissertation

zur Erlangung des akademischen Grades

Doktoringenieur (Dr.-Ing.)

von Dipl.-Ing. Andreas Wunsch
geb. am 22.05.1987 in Burg

genehmigt durch die Fakultät Maschinenbau
der Otto-von-Guericke-Universität Magdeburg

Gutachter:

Prof. Dr.-Ing. Dr. h.c. Sándor Vajna

Prof. Dr.-Ing. Dr. h.c. Tibor Bercsey

Promotionskolloquium am 14.12.2016

Andreas Wunsch

**Effizienter Einsatz von Optimierungsmethoden in der Produktentwicklung
durch dynamische Parallelisierung**

Otto-von-Guericke-Universität Magdeburg
Integrierte Produktentwicklung, Band 20

© 2017 Otto-von-Guericke-Universität Magdeburg
Lehrstuhl für Maschinenbauinformatik
Prof. Dr.-Ing. Dr. h.c. Sándor Vajna
Postfach 4120
D-39016 Magdeburg
<http://lmi.ovgu.de>

Alle Rechte vorbehalten, auch das des Nachdrucks, der Wiedergabe (Fotokopie etc.), der Speicherung in Datenverarbeitungsanlagen und der Übersetzung, auszugsweise oder vollständig.

Printed in Germany

ISBN 978-3-941016-10-1

Vorwort

Die Produktentwicklung ist die wesentliche Quelle für Innovationen eines Unternehmens sowohl beim Schaffen neuer als auch beim Anpassen vorhandener Produkte. Die Verantwortung der Produktentwicklung ist sehr hoch, da hier die wesentlichen Eigenschaften eines Produkts sowie seine Herstellungskosten und Nutzungsmöglichkeiten festgelegt werden. Gerade die Produktentwicklung sorgt aber dafür, dass der Innovationsgrad eines Produkts gesteigert, sein Nutzen erhöht, dabei gleichzeitig seine Kosten gesenkt und die Qualität sowohl des Produkts als auch der damit verbundenen Generierungs- und Anwendungsverfahren verbessert werden. Befördert und beeinflusst werden diese Ziele durch zunehmende Produktindividualisierung und -komplexität, Notwendigkeit der Ressourcenschonung und der Forderung nach erhöhter Zuverlässigkeit des Produkts. Da alle Ziele und Einflüsse sich gegenseitig auf verschiedenen Ebenen zum Teil massiv beeinflussen, lässt sich eine Verbesserung des Produkts und seiner Verfahren im weitesten Sinne nur mithilfe von multikriteriellen und multidisziplinären Optimierungsansätzen erreichen. Es ist das Ziel dieser Dissertation, deren Anwendung zu vereinfachen und effizienter zu machen.

Je mehr Einflüsse und Vorgaben bei einer Optimierung zu berücksichtigen sind, desto multikriterieller, multidisziplinärer und damit höher wird der Aufwand für Modellierung, Berechnung und Simulation des entstehenden Produkts und damit auch Aufwand und Dauer für die Optimierung. Insbesondere der zeitliche Aufwand kann verringert werden, wenn es gelingt, die Arbeiten zur parallelen Bearbeitung auf mehrere Rechner zu verteilen, ohne dass dabei die Ergebnisgüte durch diese Parallelisierung verschlechtert wird.

Die wissenschaftlichen Beiträge der vorliegenden Dissertation bestehen zunächst darin, dass alle Aktivitäten einer multikriteriellen und multidisziplinären Optimierung mithilfe des TOTE-Schemas modelliert werden. Mit dem Concurrent Engineering und dem Simultaneous Engineering werden zwei bewährte Verfahren zur Parallelisierung von Aktivitäten in der Projektarbeit auf die Organisation der Bearbeitung multidisziplinärer Optimierungsprozesse in einem Rechnercluster in den Formen von Concurrent Optimization und Simultaneous Optimization adaptiert. Für Generierung und Optimierung von Lösungsalternativen kommen Evolutionsverfahren aus der Autogenetischen Konstruktionstheorie zum Einsatz. Für die parallele Bearbeitung der Optimierung wird ein Managementsystem für einen Rechnercluster verwendet, mit dem die aktuell verfügbaren Ressourcen in den Leerlaufzeiten der Rechner in diesem Cluster gezielt genutzt und der Leerlauf bei der Bearbeitung minimiert werden, damit die maximale Leistungsfähigkeit der vorhandenen Ressourcen genutzt wird, ohne dass dabei andere Nutzer in ihrer interaktiven Arbeit behindert oder eingeschränkt werden.

Die vorliegende Dissertation leistet mit den hier vorgestellten Konzepten und Lösungen Pionierarbeit zum besseren Verständnis von Optimierungen, die nun wesentlich einfacher und günstiger werden und die weniger Zeit und keine Mehrkosten in Anspruch nehmen. Damit kann die Häufigkeit von Produktoptimierungen gesteigert werden, was letztendlich zu verbesserten Produkten führt.

Danksagung

Die vorliegende Arbeit entstand während meiner Tätigkeit als wissenschaftlicher Mitarbeiter am Lehrstuhl für Maschinenbauinformatik am Institut für Maschinenkonstruktion der Otto-von-Guericke-Universität Magdeburg. An dieser Stelle möchte ich die Gelegenheit ergreifen, mich bei denjenigen Personen zu bedanken, welche zum Erfolg dieser Arbeit beigetragen haben.

Mein besonderer Dank gilt meinem Doktorvater Herrn Prof. Dr.-Ing. Dr. h.c. Sándor Vajna, dem Inhaber des Lehrstuhls für Maschinenbauinformatik, für die Förderung und Unterstützung dieser Arbeit und darüber hinaus. Er ermöglichte die wissenschaftliche Tätigkeit in einem industrienahen Umfeld, was die Arbeit deutlich geprägt hat.

Weiterhin bedanke ich mich sehr herzlich bei Herrn Prof. Dr.-Ing. Dr. h.c. Tibor Bercsey, dem ehemaligen Inhaber des Lehrstuhls für Maschinenkonstruktion und Produktentwicklung der Technischen und Wirtschaftswissenschaftlichen Universität Budapest, für die Übernahme des Korreferats und die aufmerksame Durchsicht dieser Arbeit.

Den Kolleginnen und Kollegen des LMI danke ich für das angenehme Arbeitsklima, die Unterstützung und den fachlichen Gedankenaustausch. Ich habe mich hier stets sehr wohl gefühlt. Weiterhin bedanke ich mich bei meinen studentischen Hilfskräften für die gute Zusammenarbeit.

Abschließend danke ich all denjenigen Menschen, die mich während meines Studiums und der Promotionszeit unterstützt haben. In erster Linie sind hierbei meine Eltern zu nennen, ohne die ein erfolgreicher Abschluss in dieser Form nicht möglich gewesen wäre. Besonderer Dank gilt auch den Menschen aus meinem persönlichen Umfeld für den Austausch, die Motivation und die Unterstützung in jeglicher Form.

Magdeburg, im November 2016

Andreas Wunsch

Kurzreferat

Die vorliegende Dissertation beantwortet die Forschungsfrage der effizienten Bearbeitung von Optimierungsaufgaben in der Produktentwicklung durch den Einsatz von Parallelisierungsmethoden unter Berücksichtigung der verfügbaren Ressourcen innerhalb einer Organisation. Basierend auf Analogiebetrachtungen zum Verhalten interdisziplinärer Entwicklungsteams wird eine Methode zur effizienten Parallelisierung von Evaluationsprozessen von Optimierungen entwickelt und prototypisch in einem Framework umgesetzt. Unter den Begriffen Concurrent Optimization und Simultaneous Optimization werden die zur Parallelisierung von Produktentwicklungsprozessen eingesetzten Methoden des Concurrent Engineering und des Simultaneous Engineering auf die Anwendung in Optimierungsverfahren übertragen.

Die Parallelisierung der Evaluationsprozesse basiert auf der Dekomposition des Optimierungs- und des Evaluationsmodells und erfolgt innerhalb des entwickelten Frameworks durch dynamische Allokation. Da die Bearbeitung der Prozesselemente eines Evaluationsprozesses dabei nur auf der Verfügbarkeit und der Stabilität der benötigten Informationen und Ressourcen basiert, wird eine hohe Flexibilität in der Anwendung sowie die dynamische Reaktion auf spontane Veränderungen der Ressourcen ermöglicht. Als Ressourcen dienen die Einheiten eines Workstation-Clusters. Dies ermöglicht die Nutzung freier Rechenkapazitäten vorhandener Workstations und stellt somit eine kosteneffiziente verteilte IT-Umgebung zur Verfügung. Weiterhin werden die vorhandene Software und die verfügbaren Lizenzen berücksichtigt. Durch die prioritätenbasierte Bearbeitung der Prozesselemente kann die Effizienz einer Optimierung zusätzlich erhöht werden, da somit unnötiger Leerlauf vorhandener Ressourcen vermieden wird. Die durchgeführten Untersuchungen zeigen, dass Optimierungsverfahren im Vergleich zu Heuristiken die wesentlich effektivere und flexiblere Lösung zur Ermittlung der optimalen Prioritätswerte darstellen. Der genetische Algorithmus NSGA-II generiert dabei durchgängig die optimalen Prioritätswerte, welche in der kürzesten Laufzeit der Produktoptimierung resultieren.

Die Validierung des Frameworks erfolgt anhand von drei Fallstudien, welche charakteristische multidisziplinäre Optimierungsaufgaben aus realen Entwicklungsprojekten darstellen. Dies spiegelt ein industrienahes Umfeld wider, in dem freie Workstations während Pausen, in der Nacht oder am Wochenende als Ressourcen in einem Cluster zur Verfügung stehen können. Das entwickelte Framework kann sowohl in homogenen als auch in heterogenen Workstation-Clustern eingesetzt werden, welche kostenneutral aus einer beliebigen Anzahl in einem Netzwerk verbundener Workstations realisiert werden können. Zusätzliche Hardware wird nicht benötigt. Zudem besteht keine Notwendigkeit eines gemeinsamen Speichers, da der Datentransfer direkt zwischen den Workstations erfolgt. Jede zur Verfügung stehende Workstation kann dem Cluster schnell und einfach hinzugefügt oder aus dem Cluster entfernt werden, auch während laufenden Optimierungen. Die Workstations des Clusters stehen daher jederzeit zur interaktiven Arbeit zur Verfügung und die interaktive Arbeit wird nicht beeinträchtigt. Das entwickelte Framework eignet sich daher für die Verwendung in kleinen und mittleren Unternehmen, in denen kein oder nur eingeschränkter Zugang zu HPC-Supercomputern besteht.

Abstract

The present dissertation answers the research question of efficient execution of optimization tasks in product development by using parallelization methods and by considering the existing resources within an organization. Based on analogy observations on the behavior of interdisciplinary teams, a method for efficient parallelization of evaluation processes of optimizations was developed and implemented prototypically into a framework for distributed optimization. Using the terms of concurrent optimization and simultaneous optimization, the methods of concurrent engineering and simultaneous engineering, which are commonly used for the parallelization of product development processes, are transferred to the application of optimization procedures.

The parallelization of evaluation processes is based on the decomposition of both the optimization model and the evaluation model. Single process elements of the evaluation process are allocated dynamically within the framework. Since the processing of these process elements depends only on the availability and the stability of required information and resources, high flexibility in the application and dynamic response to spontaneous changes in the resources can be ensured. To achieve a large distributed computing environment the units of a workstation cluster are used as resources. This allows the use of free computing capacities within the workstation cluster. Thus, this cluster becomes a very cost-efficient distributed environment. Within this environment the term “resource” also covers installed software and available licenses. Furthermore, the efficiency of an optimization can be increased by using priority-based processing of the process elements, since unnecessary idling of existing resources is thus avoided. The results of this thesis show that optimization methods are more effective and more flexible for determining the optimal priority values than heuristics. Thereby, the optimal priority values can be generated best by using the genetic algorithm NSGA-II. These priority values yield to the shortest processing time of the final product optimization.

The validation of the developed framework is based on three case studies, which represent different typical multidisciplinary optimization tasks of real product development projects. Thus, the execution of these case studies reflects an industrial environment where free workstations can become available as resources in a cluster during breaks, at night, or at weekends. The framework can be used in both homogeneous and heterogeneous workstation clusters, which can be implemented cost-efficiently from an arbitrary number of workstations connected by a network. Additional hardware is not required. Since data is transferred directly between these workstations, there is no need for a shared memory environment. Even during a running optimization available workstations can be added to the cluster or removed from it very quickly. Workstations of the cluster are available for interactive work at any time. Thus, interactive work is not affected by a running optimization. Therefore, the developed framework is suitable for applications in small and medium-sized enterprises which have no or only limited access to HPC supercomputers.

Inhaltsverzeichnis

Vorwort	III
Danksagung	V
Kurzreferat	VII
Abstract	IX
Inhaltsverzeichnis	XI
Abbildungsverzeichnis	XV
Tabellenverzeichnis	XIX
Abkürzungsverzeichnis	XXI
Formelzeichenverzeichnis	XXIII
1 Einleitung	1
2 Technische Grundlagen der Produktoptimierung	7
2.1 Klassifizierung, Begriffe und Definitionen	7
2.2 Mathematische Formulierung einer Optimierungsaufgabe	16
2.3 Der Optimierungsprozess	20
2.4 Designvariablen- und Lösungsraum im Optimierungsprozess	21
2.5 Optimierungsmethoden	23
2.5.1 Parameteroptimierung	24
2.5.2 Topologieoptimierung	30
2.5.3 Formoptimierung	34
2.5.4 Freie Dimensionierung	36
2.6 Optimierungsverfahren	37
2.6.1 Deterministische Optimierungsverfahren	38
2.6.2 Stochastische Optimierungsverfahren	42
2.6.2.1 Evolutionäre und genetische Algorithmen	44
2.6.2.2 Partikelschwarmverfahren (Particle Swarm)	53
2.6.2.3 Simuliertes Ausglühen (Simulated Annealing)	54
2.6.2.4 Abschließende Bemerkungen	55
2.6.3 Hybride Optimierungsverfahren und Weitere	55
2.7 Explorative Untersuchung des Lösungsraumes	56
2.7.1 Design of Experiments	57
2.7.2 Generierung von Metamodellen	63
2.8 Optimierungsstrategien	70
2.8.1 Multikriterielle Optimierung	72
2.8.1.1 Zielgewichtung	76
2.8.1.2 Abstandsfunktion	78
2.8.1.3 Restriktionsformulierung	80

2.8.1.4	Multikriterielle genetische Algorithmen	83
2.8.1.5	Abschließende Bemerkungen	84
2.8.2	Multidisziplinäre Optimierung	85
2.8.3	Kombination von Optimierungsmethoden und -verfahren	87
3	Analogiebetrachtungen von Optimierungsverfahren und Produktentwicklung	91
3.1	Biologisch inspirierte Programmierung	91
3.2	Autogenetische Konstruktionstheorie	95
3.2.1	Produktmodell	96
3.2.2	Lösungsraum	103
3.2.3	Anwendung der AKT in den frühen Phasen der Produktentwicklung	107
4	Effizienzsteigerung von Optimierungsverfahren durch Parallelisierung	111
4.1	Parallelisierung von Optimierungsverfahren durch Modelldekomposition .	111
4.2	Anwendung von Parallelisierungsmethoden in der Optimierung	117
4.3	Concurrent und Simultaneous Optimization als Ansatz zur Effizienzsteigerung	121
4.4	Verteiltes Rechnen	126
4.4.1	Cluster-Computing	127
4.4.2	Auswahl eines Cluster-Management-Systems	129
4.5	Prioritätenbasierte Bearbeitung	131
4.6	Abschließende Bemerkungen	136
5	Prototypische Umsetzung	137
5.1	Anforderungen	137
5.2	Konzeptionierung	140
5.3	Clusterintegration und -konfiguration	143
5.3.1	Definition und Übermittlung einzelner Jobs	143
5.3.2	Definition und Übermittlung zusammenhängender Jobs	148
5.3.3	Monitoring von Ressourcen und Jobs	149
5.3.4	Priorisierung von Jobs	153
5.3.5	Fehlerbehandlung bei Lizenzfehlern	154
5.3.6	Definition und Verwaltung von Ressourcen	156
5.3.7	Priorisierung von Ressourcen	159
5.3.8	Energieverwaltung des Clusters	160
5.3.9	Konfiguration externer Software	162
5.4	Optimierungssystem DAG2OPT	163
5.4.1	Definition des Optimierungs- und Evaluationsmodells	165
5.4.2	Optimierung der Prozessprioritäten	168
5.5	Validierung durch Fallstudien	179
5.5.1	Fallstudie 1: Idealierte Optimierung einer Interieurkomponente .	179
5.5.2	Fallstudie 2: DoE-Studie eines Kurbelarms	182
5.5.3	Fallstudie 3: Optimierung eines Hebelmechanismus	187
5.6	Handlungsempfehlungen zur effizienten Durchführung einer Optimierung .	190
6	Zusammenfassung und Ausblick	193
	Literaturverzeichnis	199

Anhang	A-1
A.1 Parametrische CAD-Modellierung	A-1
A.2 Evolutionäre Algorithmen	A-2
A.2.1 Binäre Codierung von Designvariablen	A-2
A.2.2 Beispiele evolutionärer Algorithmen	A-3
A.3 Scheduling von Berechnungsjobs	A-4
A.3.1 Prioritätenbasierte Bearbeitung	A-4
A.3.2 Scheduling-Verfahren	A-5
A.4 Cluster-Management-Systeme	A-7
A.5 Grafische Modellierungssoftware	A-12
A.6 Optimierung der Prozessprioritäten in DAG2OPT	A-14
A.6.1 Overhead-Zeit von HTCCondor	A-14
A.6.2 Allokationsverhalten von HTCCondor	A-15
A.6.3 Simulation der Allokation von HTCCondor	A-16
A.6.4 Anteil optimaler Prioritätswerte für das Testszenario 2	A-17
A.6.5 Testszenario zur Überprüfung der Funktionsweise	A-18
A.6.6 Laufzeitapproximation der Optimierung	A-19
A.7 Fallstudien	A-24
A.7.1 Laufzeitverteilung der Prozesselemente in Fallstudie 3	A-24
A.7.2 Optimierungsergebnisse von Fallstudie 3	A-26
A.8 Verwendeter Quellcode	A-32
A.8.1 Konfigurationsdatei des verwaltenden Rechners des Clusters	A-32
A.8.2 Konfigurationsdatei des ausführenden Rechners des Clusters	A-33
A.8.3 Jobdefinitionsdatei	A-35
A.8.4 DAGMan-Eingabedatei	A-36
A.8.5 Post-Skript zur Fehlerbehandlung von Lizenzfehlern	A-36

Abbildungsverzeichnis

Abb. 1.1	Attribute des IDE zur vollständigen Beschreibung des Verhaltens eines Produktes	1
Abb. 2.1	Kategorien der Strukturoptimierung	8
Abb. 2.2	Klassifizierung von Optimierungsdisziplinen	9
Abb. 2.3	Klassifizierung nach Strategieebene, Suchrichtungserzeuger und Schrittweite	10
Abb. 2.4	Hierarchische Unterteilung von Optimierungsmethoden	11
Abb. 2.5	Klassifizierung von Optimierungsmethoden und Vorgehen zur Festlegung der Optimierungsstrategie	13
Abb. 2.6	Lokales und globales Minimum eindimensionaler Zielfunktionen	17
Abb. 2.7	Definition der Konvexität einer Funktion	18
Abb. 2.8	Konvexität des zulässigen Bereichs einer Funktion	18
Abb. 2.9	Konvexität einer Menge im Designvariablenraum	19
Abb. 2.10	Ursprüngliche Version des TOTE-Schemas	20
Abb. 2.11	TOTE-Schema als Prozessschaubild	21
Abb. 2.12	Abbildung des Lösungsraumes aus dem Designvariablenraum und Rückführung der Ergebnisse	22
Abb. 2.13	Schematische Darstellung der CAD/CAE-Integration	25
Abb. 2.14	Parametrisierte Steuer- und Konturskizze eines CAD-Modells unter Verwendung von NURBS	27
Abb. 2.15	Topologieoptimierung eines biegebeanspruchten Hakens	31
Abb. 2.16	Topologieoptimierung eines biegebeanspruchten Balkens	32
Abb. 2.17	Teilautomatische Generierung des CAD-Modells aus dem Ergebnis einer Topologieoptimierung	33
Abb. 2.18	Formoptimierung einer zugbeanspruchten Platte	35
Abb. 2.19	Sickenoptimierung einer Ölwanne	36
Abb. 2.20	Ergebnisse der Topologieoptimierung und der freien Dimensionierung eines Kragträgers	37
Abb. 2.21	Topologieoptimierung eines Kragträgers unter Verwendung eines zehnschichtigen Volumenmodells	37
Abb. 2.22	Grundstruktur eines einfachen evolutionären Algorithmus	45
Abb. 2.23	Fitnesszuweisung bei linearem und nichtlinearem Ranking	47
Abb. 2.24	Rouletteauswahl und stochastisch universelle Selektion	49
Abb. 2.25	Funktionsweise der genetischen Operatoren Rekombination und Mutation	50
Abb. 2.26	Reduzierung der Laufzeit einer Optimierung durch den Einsatz von Metamodellen	57
Abb. 2.27	Vollfaktorielle Versuchspläne	59
Abb. 2.28	Teilfaktorielle Versuchspläne	60
Abb. 2.29	Box-Behnken-Versuchsplan	61
Abb. 2.30	Central-Composite-Versuchsplan	62
Abb. 2.31	Monte-Carlo-Versuchsplan mit 100 Stützstellen	63
Abb. 2.32	Latin-Hypercube-Versuchsplan	63
Abb. 2.33	Approximation einer Sinusfunktion durch ein Polynom 2. Grades	65

Abb. 2.34	Approximation einer Sinusfunktion durch Kriging-Modelle unterschiedlicher Gewichtung	66
Abb. 2.35	Approximation einer Sinusfunktion durch unterschiedliche Basisfunktionen	68
Abb. 2.36	Approximation einer Sinusfunktion durch Neuronale Netzwerke unterschiedlicher Startgewichte und Neuronenanzahl	69
Abb. 2.37	Abbildung des multikriteriellen Lösungsraumes aus den Zielkriterien . .	73
Abb. 2.38	Pareto-Fronten im Lösungsraum in Abhängigkeit vom Optimierungsziel	73
Abb. 2.39	Charakteristische Pareto-Fronten im Lösungsraum	74
Abb. 2.40	Dominanz von Lösungen im Lösungsraum	75
Abb. 2.41	Formulierung der Zielfunktion durch Zielgewichtung	77
Abb. 2.42	Lösungen bei verschiedenen Anspruchsniveaus	79
Abb. 2.43	Restriktionsformulierung der Zielfunktion	81
Abb. 2.44	Externe und interne Straffunktionen	82
Abb. 2.45	Rangzuweisung durch nicht dominierte Sortierung	84
Abb. 3.1	Produktmodell der AKT	97
Abb. 3.2	Grenzen und Elemente des Lösungsraumes der AKT	104
Abb. 3.3	Schematische Entwicklung des Lösungsraumes der AKT bei der Produktentwicklung	105
Abb. 3.4	Benutzungsoberfläche von Morphix zur interaktiven Konzeptevaluation	108
Abb. 4.1	Verlauf von Speedup und Effizienz bei der parallelen Verarbeitung . . .	113
Abb. 4.2	Prinzipieller Aufbau des Master-Worker-Schemas	114
Abb. 4.3	Sequentielle Verarbeitung von Optimierungsprozessen	116
Abb. 4.4	Parallele Verarbeitung von Optimierungsprozessen	116
Abb. 4.5	Parallelisierung genetischer Algorithmen	118
Abb. 4.6	Zusammenhang von Effektivität, Ressourcen und Laufzeit einer Optimierung	121
Abb. 4.7	Concurrent Engineering und Simultaneous Engineering	123
Abb. 4.8	Grundlegender Ablauf der Concurrent und Simultaneous Optimization-Methode	126
Abb. 4.9	Ergebnis des binären Vergleiches windows-basierter Cluster-Management-Systeme	130
Abb. 4.10	Übermittlung von Jobs an die Warteschlange	131
Abb. 4.11	Reduzierung der Laufzeit durch prioritätenbasierte Bearbeitung bei der Verwendung von zwei Berechnungsslots	132
Abb. 4.12	Klassifizierung von Scheduling-Verfahren	134
Abb. 5.1	Aktivitätsdiagramm zur Durchführung einer Optimierung	141
Abb. 5.2	Aufgaben und Interaktionen der in das Framework integrierten Systeme	142
Abb. 5.3	Hauptkomponenten eines HTCondor-Clusters	144
Abb. 5.4	Wesentliche Dienste der Hauptkomponenten eines HTCondor-Clusters .	145
Abb. 5.5	DAG-Beschreibung eines Workflows	148
Abb. 5.6	Hauptoberfläche des Programms Condor GUI	152
Abb. 5.7	Verwendung von Pre- und Postskripten in einem DAG	155
Abb. 5.8	LSP-Fix zur Überprüfung der LSPs eines Rechners	163
Abb. 5.9	Benutzungsoberfläche des Optimierungssystems DAG2OPT	164
Abb. 5.10	Tabellendarstellung der Evaluationsprozesse in DAG2OPT	165

Abb. 5.11	Definition der Designvariablen in DAG2OPT	166
Abb. 5.12	Definition der Zielfunktion in DAG2OPT	167
Abb. 5.13	Testscenarien für die Bestimmung des Verfahrens zur Optimierung der Prozessprioritäten	168
Abb. 5.14	Vergleich von simulierter und realer Laufzeit für Testscenario 1 ($n_s = 4$)	169
Abb. 5.15	Vergleich von simulierter und realer Laufzeit für Testscenario 2 ($n_s = 4$)	171
Abb. 5.16	Aktivitätsdiagramm zur Optimierung der Prozessprioritäten	172
Abb. 5.17	Vergleich verschiedener Optimierungsalgorithmen zur Optimierung der Prozessprioritäten für Testscenario 2 ($n_p = 4, n_s = 4$)	174
Abb. 5.18	Vergleich verschiedener Optimierungsalgorithmen zur Optimierung der Prozessprioritäten für Testscenario 2 ($n_p = 15, n_s = 4$)	175
Abb. 5.19	Laufzeitreduzierung durch Verwendung optimierter Prozessprioritäten im Testscenario 3 ($n_p = 11, n_s = 4$)	176
Abb. 5.20	Ergebnis der Optimierung der Prozessprioritäten in DAG2OPT für Testscenario 2	178
Abb. 5.21	Prozesskette der Optimierung	179
Abb. 5.22	Laufzeiten der Prozesselemente der Evaluation	180
Abb. 5.23	DAG des Evaluationsprozesses der Fallstudie 2	183
Abb. 5.24	Reale und idealisierte Laufzeit verschiedener Parallelisierungsmethoden	184
Abb. 5.25	Speedup und Effizienz der vollständig parallelen Verarbeitung	186
Abb. 5.26	Vergleich der Ergebnisse der durchgeführten Optimierungen	188
Abb. 5.27	Evaluationen der Optimierungen	188
Abb. 5.28	Evaluationen von Optimierung 2 bei einer erneuten Durchführung . . .	189
Abb. A.1	Flexibilität eines parametrisierten CAD-Modells unter Verwendung ei- ner Steuerskizze und NURBS	A-1
Abb. A.2	Reduzierung der Laufzeit durch prioritätenbasierte Bearbeitung bei der Verwendung von drei Berechnungsslots	A-4
Abb. A.3	Experimentelle Ermittlung der durchschnittlichen Overhead-Zeit von HTCondor	A-14
Abb. A.4	Überschreibung der Prozessprioritäten eines DAG durch HTCondor . .	A-15
Abb. A.5	Aktivitätsdiagramm der Allokationssimulation von HTCondor	A-16
Abb. A.6	Prozentualer Anteil optimaler Prioritätswerte für das Testscenario 2 . .	A-17
Abb. A.7	Testscenario 3 zur Überprüfung der Funktionsweise des Werkzeugs zur Optimierung der Prozessprioritäten ($n_s = 4$)	A-18
Abb. A.8	Untersuchte Kombinationen der Eingabeparameter zur Laufzeitunter- suchung der Allokationssimulation	A-19
Abb. A.9	Laufzeit der Allokationssimulation für definierte Werte der Slotanzahl .	A-20
Abb. A.10	Laufzeit der Allokationssimulation für reduzierte Werte der Slotanzahl .	A-20
Abb. A.11	Laufzeit der Allokationssimulation für definierte Werte der Prozess- elementanzahl	A-21
Abb. A.12	Laufzeit der Allokationssimulation für definierte Größen der Paralleli- sierungsdomäne	A-22
Abb. A.13	Mindestens benötigte Evaluationen bei der Optimierung der Prozess- prioritäten	A-23
Abb. A.14	DAG des Evaluationsprozesses der Fallstudie 3	A-24
Abb. A.15	Laufzeitverteilung des Prozesselementes A	A-24
Abb. A.16	Laufzeitverteilung des Prozesselementes B	A-25
Abb. A.17	Laufzeitverteilung des Prozesselementes C	A-25

Abb. A.18	Gegenüberstellung der Zielkriterien der Individuen des zweiten Optimierungslaufs	A-27
Abb. A.19	Aus der Menge Pareto-optimaler Lösungen gewählte Variante des Hebelmechanismus	A-27
Abb. A.20	Verlauf des Zielfunktionswertes und Rang der 50 besten Individuen einer Generation des zweiten Optimierungslaufs	A-28
Abb. A.21	Polardiagramm der Designvariablen der 100 besten Varianten bezogen auf den Zielfunktionswert	A-30
Abb. A.22	Polardiagramm der Designvariablen der 100 besten Varianten bezogen auf die Zustandsvariable Masse	A-30
Abb. A.23	Polardiagramm der Designvariablen der 100 besten Varianten bezogen auf die Zustandsvariable max. Verschiebung	A-31
Abb. A.24	Polardiagramm der Designvariablen der 100 besten Varianten bezogen auf die Zustandsvariable max. Spannung	A-31

Tabellenverzeichnis

Tab. 2.1	Unterscheidung von Optimierungsmethoden	23
Tab. 2.2	Arten der Parametrisierung zur Parameteroptimierung	26
Tab. 2.3	Bewertung eindimensionaler Optimierungsverfahren	41
Tab. 2.4	Bewertung deterministischer Optimierungsverfahren ohne Restriktionen	41
Tab. 2.5	Lineares Ranking und Normierung des Fitnesswertes	48
Tab. 2.6	Charakteristik stochastischer Optimierungsverfahren	55
Tab. 2.7	Charakteristik und Stützstellen etablierter Versuchspläne	58
Tab. 3.1	Analogien der biologischen Evolution und evolutionärer Algorithmen	92
Tab. 3.2	Unterstützung von Tätigkeiten der Produktentwicklung durch evolutionäre Operatoren	95
Tab. 5.1	Laufzeiten und verwendete Ressourcen unter der Verwendung verschiedener Parallelisierungsmethoden	181
Tab. A.1	Vergleich von Binärcodierung und Cray-Codierung	A-2
Tab. A.2	Übersicht evolutionärer und genetischer Algorithmen	A-3
Tab. A.3	Verfahren für das Scheduling von Berechnungsjobs	A-5
Tab. A.4	Anforderungsliste zur Recherche und Auswahl eines Cluster-Management-Systems	A-7
Tab. A.5	Windows-basierte Cluster-Management-Systeme	A-9
Tab. A.6	Binärer Vergleich der Funktionalität ausgewählter Cluster-Management-Systeme	A-11
Tab. A.7	Binärer Vergleich der Benutzbarkeit ausgewählter Cluster-Management-Systeme	A-11
Tab. A.8	Binärer Vergleich der Umsetzbarkeit ausgewählter Cluster-Management-Systeme	A-11
Tab. A.9	Windows-basierte grafische Modellierungssoftware	A-12
Tab. A.10	Designvariablen und Werte der Laufzeitapproximation	A-19

Abkürzungsverzeichnis

AKT	Autogenetische Konstruktionstheorie
API	Application Programming Interface (Anwendungsprogrammierschnittstelle)
ASCII	American Standard Code for Information Interchange, ANSI X3.4 (Amerikanischer Standard-Code für Informationsaustausch)
BFGS	Broyden-Fletcher-Goldfarb-Shanno-Verfahren
CAD	Computer-Aided Design
CAE	Computer-Aided Engineering
CAM	Computer-Aided Manufacturing
CAx	Computer-Aided x (Zusammenfassung computerunterstützter Systeme)
CCC	Central-Composite-Circumscribed (Außenkreisbegrenzter Versuchsplan)
CCF	Central-Composite-Faced (Flächenbegrenzter Versuchsplan)
CCI	Central-Composite-Inscribed (Innenkreisbegrenzter Versuchsplan)
CE	Concurrent Engineering
CFD	Computational Fluid Dynamics (Numerische Strömungsmechanik)
CO	Concurrent Optimization
CPU	Central Processing Unit (Zentraler Prozessor eines Computers)
DAG	Directed Acyclic Graph (Gerichteter azyklischer Graph)
DFP	Davidon-Fletcher-Powell-Verfahren
DLL	Dynamic Link Library (Dynamische Programmbibliothek)
DoE	Design of Experiments (Statistische Versuchsplanung)
EA	Evolutionärer Algorithmus
ERP	Enterprise-Resource-Planning
ES	Evolutionsstrategie
FE	Finite Elemente
FEM	Finite-Elemente-Methode
FIFO	First In - First Out (Reihenfolgebasierte Bearbeitung)
FLOPS	Floating Point Operations Per Second (Gleitkommaoperationen pro Sekunde)
GA	Genetischer Algorithmus
GPU	Graphics Processing Unit (Grafikprozessor)

GUI	Graphical User Interface (Grafische Benutzungsoberfläche)
HPC	High-Performance Computing
HTC	High-Throughput Computing
IDE	Integrated Design Engineering
IP	Internetprotokoll
IT	Informationstechnik
LSP	Layered Service Provider
MFD	Method of Feasible Directions (Methode der zulässigen Richtungen)
MG	Modellgenerierung
MMFD	Modified Method of Feasible Directions (Modifizierte Methode der zulässigen Richtungen)
NURBS	Non-Uniform Rational B-Spline
PP	Produktparameter
RSM	Response Surface Model (Antwortflächenbasiertes Metamodell)
SA	Simulated Annealing (Simuliertes Ausglühen)
SE	Simultaneous Engineering
SIMP	Solid Isotropic Material with Penalization
SKO	Soft Kill Option
SO	Simultaneous Optimization
TCP	Transmission Control Protocol (Übertragungssteuerungsprotokoll)
TOTE	Test Operate Test Exit (Iterativer Prozess zur Problemlösung)
TSP	Traveling Salesman Problem (Problem des Handlungsreisenden)
UML	Unified Modeling Language (Vereinheitlichte Modellierungssprache)
XML	Extensible Markup Language (Erweiterbare Auszeichnungssprache)

Formelzeichenverzeichnis

Formelzeichen	Einheit	Bezeichnung
E_p	–	Effizienz paralleler Verarbeitung
f	–	Zielfunktion
\mathbf{f}	–	Vektor der Zielfunktionen
\tilde{f}	–	Ersatzzielfunktion
\hat{f}	–	Approximierte Zielfunktion
f_{Fit}	–	Fitness
F	–	Lösungsraum, Menge zulässiger Lösungen
g	–	Ungleichheitsrestriktion
\mathbf{g}	–	Ungleichheitsrestriktionsvektor
h	–	Gleichheitsrestriktion
\mathbf{h}	–	Gleichheitsrestriktionsvektor
I	–	Individuum eines evolutionären Algorithmus
K_i	–	Steifigkeitsmatrix eines finiten Elementes
\underline{K}_i	–	Bestrafte Steifigkeitsmatrix eines finiten Elementes
l	–	Stufen/Level einer vollfaktoriellen DoE
m	kg	Masse
M	–	Menge
n	–	Anzahl der Designvariablen
n_e	–	Anzahl der Prozesselemente eines DAG
n_f	–	Anzahl der Zielkriterien
n_g	–	Anzahl der Ungleichheitsrestriktionen
n_h	–	Anzahl der Gleichheitsrestriktionen
n_I	–	Populationsgröße eines EA bzw. GA
n_p	–	Größe der Parallelisierungsdomäne
n_s	–	Anzahl der Slots zur parallelen Verarbeitung
p	–	Straffunktion, Exponent der Abstandsfunktion
p_k	–	Glättungsfaktor beim Kriging
p_s	–	Selektionsdruck

P	–	Menge Pareto-optimaler Lösungen
q	–	Skalierungsfaktor der Straffunktion
r	–	Reduktionsstufe teilfaktorieller Versuchspläne
r_I	–	Rang von Individuen
\mathbb{R}^n	–	n -dimensionale Menge der reellen Zahlen
\mathbb{R}^+	–	Menge der positiven reellen Zahlen
S_p	–	Speedup bei paralleler Verarbeitung
t	s	Laufzeit
t_i	s	Laufzeit einzelner Prozesselemente
t_{over}	s	Overhead-Zeit
t_p	s	Parallele Laufzeit
t_{seq}	s	Sequentielle Laufzeit
w	–	Gewichtungsfaktor
x	–	Designvariable
x^*	–	Optimierte Designvariable
x_a	–	Wert einer Designvariable
x_b	–	Wert einer Designvariable
x_c	–	Wert einer Designvariable
x^l	–	Untere Grenze einer Designvariable
x^u	–	Obere Grenze einer Designvariable
\mathbf{x}	–	Vektor der Designvariablen
\mathbf{x}^*	–	Vektor der optimierten Designvariablen
\mathbf{x}_a	–	Wertevektor der Designvariablen
\mathbf{x}_b	–	Wertevektor der Designvariablen
\mathbf{x}_c	–	Wertevektor der Designvariablen
\mathbf{x}^l	–	Unterer Grenzwertevektor der Designvariablen
\mathbf{x}^u	–	Oberer Grenzwertevektor der Designvariablen
X	–	Designvariablenraum, Menge zulässiger Designvariablenwerte
\mathbf{y}	–	Konvexkombinationen
\bar{y}	–	Anspruchsniveau der Zielfunktion
$\bar{\mathbf{y}}$	–	Vektor der Anspruchsniveaus

\hat{y}	–	Antwortfläche
u	mm	Verschiebung
ϵ	–	Fehler in der Approximation der Zielfunktion
θ	–	Gewichtungsfaktor beim Kriging
Θ	–	Laufvariable zur Bestimmung der Konvexität
μ	–	Mittelwert der Stützstellen beim Kriging
ρ_i	–	Normierte Dichte eines finiten Elementes
σ	MPa	Mechanische Spannung

1 Einleitung

In der Produktentwicklung spielt die Effizienz in vielen Aspekten eine entscheidende Rolle. In erster Linie soll das Produkt selbst während der Nutzungsphase effizient sein. Aber auch bei der Produktion und in der Phase nach der Nutzung, in der die Demontage, das Recycling und die Entsorgung des Produktes erfolgen, ist die Effizienz der involvierten Prozesse von Bedeutung. Neben der Produktion muss aus Sicht eines Unternehmens vor allem auch der Produktentwicklungsprozess effizient gestaltet sein. Engpässe und Leerlauf sollten vermieden und die zur Verfügung stehenden Ressourcen in der bestmöglichen Art und Weise genutzt werden.

Getrieben durch die steigende Kundenindividualität, Ressourcenschonung und Zuverlässigkeit eines Produktes, ist die effiziente und ressourcenschonende Entwicklung innovativer Produkte von entscheidender Bedeutung für die Wettbewerbsfähigkeit von Unternehmen [BMB12]. Dies wird zudem durch die sich stetig verkürzenden Produktlebenszyklen von Produkten verdeutlicht, wodurch Unternehmen gezwungen sind, Innovationen immer schneller in den Markt einzuführen und neue Marktimpulse zu setzen, um die eigene Wettbewerbsfähigkeit zu gewährleisten [Gro14]. Die mit den kürzer werdenden Produktlebenszyklen verbundenen kürzeren Entwicklungs- und Innovationszeiten sowie stetig steigende Qualitätsansprüche der Kunden erfordern somit gleichermaßen effektive und effiziente Prozesse in der Produktentwicklung [YPY10].

Die damit verbundene Komplexität der Produktentwicklung lässt sich nur beherrschen, wenn den prozessbeteiligten Personen alle entwicklungsrelevanten Informationen bereitstehen und die Entwicklung einer effizienten Lösung gemeinsam in einem interdisziplinären Team erfolgt [BMB12]. Dabei müssen das zu entwickelnde Produkt sowie der Produktlebenszyklus ganzheitlich betrachtet werden.

Diesen Ansatz verfolgt das Integrated Design Engineering (IDE). Das Verhalten des zu entwickelnden Produktes wird dabei über den gesamten Lebenszyklus durch Attribute beschrieben, welche stets gleichwertig sind und sich gegenseitig bedingen [Vaj14]. Die Attribute eines Produktes und deren Wechselwirkungen sind in Abbildung 1.1 dargestellt. Eine detaillierte Beschreibung der einzelnen Attribute wird in [Vaj14] gegeben.

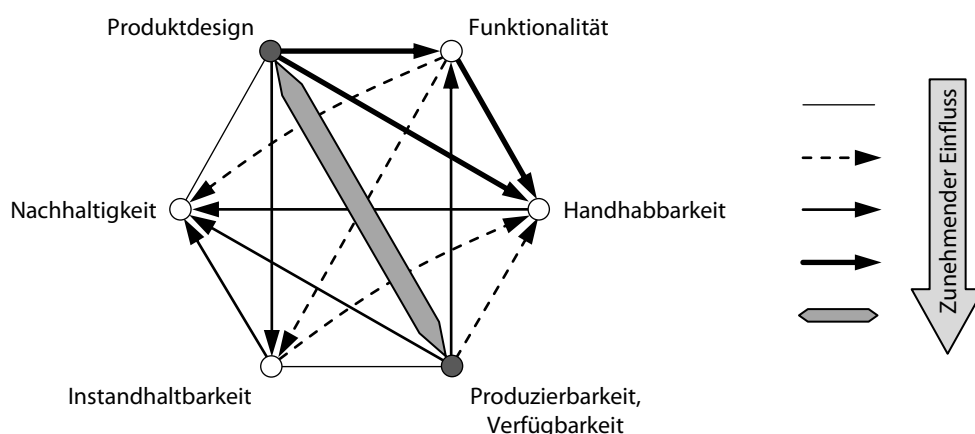


Abbildung 1.1: Attribute des IDE zur vollständigen Beschreibung des Verhaltens eines Produktes, nach [Vaj15]

Die Wechselwirkungen und gegenseitigen Einflüsse der Attribute sind unterschiedlich stark ausgeprägt. Den größten Einfluss besitzen die Attribute Produktdesign und Produzierbarkeit bzw. Verfügbarkeit, da diese durch das Entwicklungsteam direkt festgelegt werden können. Durch Variation dieser Attribute kann Einfluss auf die Ausprägung der anderen Attribute genommen werden [Vaj15]. So kann z. B. durch eine Änderung des Produktdesigns in Form einer Variation der Gestalt oder des Materials die Funktionalität oder die Produzierbarkeit des Produktes beeinflusst werden. Aufgrund dieses Einflusses sind die Gestalt und das Material auch für die fertigungsgerechte und zielkostenorientierte Entwicklung eines Produktes von entscheidender Bedeutung [EKLM14].

Bei der Entwicklung eines Produktes erlaubt der Einsatz von Simulationsverfahren dem Produktentwickler, bereits in frühen Entwicklungsphasen Aussagen über die Eigenschaften und das Verhalten eines Produktes zu treffen und gezielt die Gestalt oder das Material des Produktes zur Verbesserung dieser Eigenschaften zu verändern. Die Methoden des Computer-Aided Engineering (CAE) sind hierbei bereits fest in die verschiedenen Phasen der Produktentwicklung integriert und ein wesentlicher Erfolgsfaktor für einen effizienten, transparenten und qualitativ hochwertigen Produktentwicklungsprozess [SR08]. Dies reduziert die Anzahl physischer Prototypen und führt zu einer Kosten- und Zeiteinsparung. Der Einsatz von CAE-Methoden dient somit dazu, die eingangs beschriebenen Herausforderungen der Produktentwicklung durch die frühzeitige Ermittlung entwicklungsrelevanter Informationen zu beherrschen. Dies führt in den meisten Unternehmen zu einer ansteigenden Nutzung virtueller Prototypen [Cou08a].

Zudem können anhand des virtuellen Prototyps frühzeitig Optimierungsmethoden genutzt werden, um die Produkteigenschaften gezielt zu verbessern. In der Produktentwicklung sind dabei multikriterielle und multidisziplinäre Optimierungsstrategien von besonderer Bedeutung, da bei der Entwicklung eines Produktes in der Regel mehrere Anforderungen berücksichtigt werden müssen, welche auch gegenläufigen Charakter aufweisen sowie das Zusammenspiel verschiedener Fachdisziplinen erfordern können. Der Einsatz kann dabei auf verschiedenen Detaillierungsstufen und in verschiedenen Phasen der Entwicklung erfolgen. Durch die Anwendung von Optimierungsmethoden wird zum einen die Effektivität des Produktentwicklungsprozesses erhöht, da insbesondere bei multikriteriellen Problemstellungen die Lösungssuche durch einen Algorithmus zielgerichteter und systematischer erfolgt als die rein manuelle Lösungssuche und somit in der Regel bessere Lösungen gefunden werden. Zum anderen erhöht sich die Effizienz der Produktentwicklung, da die Evaluationsprozesse der Optimierung vollständig in digitaler Form vorliegen und automatisiert durchgeführt werden, was insbesondere bei multidisziplinären Problemen deutlich effizienter ist als die manuelle Bearbeitung.

Durch die Verwendung paralleler Berechnungsmethoden (Parallel Computing) kann der mit dem Einsatz von CAE- und Optimierungsmethoden verbundene numerische Aufwand auf verschiedene Ressourcen verteilt werden, wodurch sich die Laufzeit reduzieren lässt. Dabei sind konventionelle Parallelisierungsmethoden meist nur wirklich effizient, wenn eine große Anzahl ausführender Einheiten zur Verfügung stehen. Dies wird als High-Performance Computing (HPC) bezeichnet. Entscheidend bei der Umsetzung dieses Ansatzes sind die damit verbundenen Kosten [Cou14]. Aufgrund der hohen Anschaffungskosten von HPC-Supercomputern stehen diese oft nur großen Konzernen und Forschungsinstituten zur Verfügung [Gen15].

Laut einer Studie des Council on Competitiveness [Cou08b] nutzen 97% der Unternehmen konventionelle Desktop-PCs (z. B. Workstations oder Laptops) zur Verifikati-

on von Produkten anhand virtueller Prototypen. Lastspitzen, in denen die vorhandenen IT-Ressourcen zur Bearbeitung richtungsweisender Projekte nicht ausreichen, treten lediglich zwei bis dreimal pro Jahr auf [MC16]. Diese Lastspitzen lassen sich z. B. durch sog. HPC-Cloud-Services kompensieren. Hierbei werden die Berechnungsmodelle über das Internet an HPC-Dienstleistungsunternehmen übermittelt, welche die numerische Berechnung durchführen. Nach der Durchführung der Berechnung werden die Ergebnisse zurück an das beauftragende Unternehmen gesendet [Gen15], [MPR15]. Ein Risiko von Cloud-Services stellt jedoch der mögliche Verlust von Unternehmensgeheimnissen und der Missbrauch von Endkundendaten dar [HC15]. Insbesondere im Rahmen von Optimierungen, bei der die Modelle aufgrund der Parametrisierung neben der reinen Geometrie zusätzliche Informationen enthalten, die z. B. Rückschlüsse auf die technologische Expertise eines Unternehmens ermöglichen, haben sich Cloud-Services noch nicht etabliert.

An dieser Stelle setzt die vorliegende Arbeit an und gibt eine Antwort auf die Forschungsfrage, wie Parallelisierungsmethoden genutzt werden können, um Optimierungsprobleme hinsichtlich der verfügbaren Ressourcen innerhalb einer Organisation effizient zu bearbeiten. Dies beinhaltet die Berücksichtigung der Flexibilität in der Nutzung sowie die Skalierbarkeit der für die Optimierung benötigten Ressourcen.

Aufgrund der Parallelisierung ist der gesamte Optimierungsprozess anfällig für Engpässe, sei es durch den Mangel an Informationen, z. B. benötigte Eingabedaten, oder den Mangel an verfügbaren Ressourcen, z. B. Hardware, Software und Softwarelizenzen. Eine fehlende Information oder Ressource kann somit den gesamten Prozess aufhalten und den durch die Parallelisierung gewonnenen Vorteil mindern oder gänzlich zunichte machen. Im Rahmen der Arbeit stehen daher auch die flexible Verarbeitung vorhandener Informationen sowie die flexible und robuste Berücksichtigung relevanter Ressourcen im Fokus, welche sich zudem im Verlauf einer Optimierung ändern können. Leerlauf von Ressourcen aufgrund von Engpässen soll dabei in jedem Fall vermieden werden. Zur Kompensierung von Ressourcenausfällen und spontaner Bereitstellung zusätzlicher Ressourcen wird ein dezentraler Ansatz gewählt, da dieser deutlich ausfalltoleranter als zentrale Verfahren ist und die Skalierbarkeit des Systems ermöglicht [BBKS15]. Der in der vorliegenden Arbeit verfolgte Ansatz zur effizienten Parallelisierung kann somit als eine organisationsinterne Cloud-Umgebung durch Nutzung der vorhandenen dezentralen Ressourcen angesehen werden. Dabei bleiben alle verwendeten Modelle innerhalb der Organisation. Aus Gründen der angestrebten Skalierbarkeit soll auch die prinzipielle Möglichkeit gegeben werden, das System durch interne und externe Ressourcen HPC-Ressourcen flexibel zu erweitern.

Die vorliegende Arbeit orientiert sich an dem in [DO99] beschriebenen Ansatz zur Bearbeitung von Forschungsfragen in der Produktentwicklung. Neben der Identifizierung der Problemstellung beinhaltet die Arbeit daher auch die Verfolgung der Hypothese, dass durch die prioritätenbasierte Bearbeitung von Berechnungsaufgaben eine Effizienzsteigerung des Optimierungsprozesses ermöglicht wird. Die hierzu erarbeitete Lösung wird anhand von Testszenarien und Fallstudien, welche multidisziplinäre Optimierungen realer Entwicklungsprojekte darstellen, validiert.

Die effiziente Durchführung von Optimierungen erfordert fundierte Kenntnisse der Funktionsweise von Optimierungsmethoden und -verfahren. Daher erfolgt in der vorliegenden Arbeit zunächst eine ausführliche Beschreibung der technischen Grundlagen der Produktoptimierung. Zu Beginn werden verschiedene Klassifizierungen von Optimierungsmethoden und -verfahren aus der Literatur analysiert und zu einer Gesamtdarstellung synthetisiert. Diese ermöglicht eine vollständige Einordnung der in der Produktentwicklung

etablierten Optimierungsmethoden, -verfahren und -algorithmen. Zudem bildet die Klassifizierung die Grundlage für die Definition der wichtigsten Begriffe der Arbeit und stellt gleichermaßen ein prinzipielles Vorgehensmodell für die Festlegung von Optimierungsstrategien zur Bearbeitung von Optimierungsaufgaben dar. Bei der Beschreibung der weiteren technischen Grundlagen werden multikriterielle und multidisziplinäre Optimierungsstrategien hervorgehoben, da diese in der Produktentwicklung aufgrund der unterschiedlichen Anforderungen an ein Produkt von besonderer Bedeutung sind. Auch die Kombination von Optimierungsmethoden und -verfahren wird betrachtet, da dadurch die Effektivität und Effizienz von Optimierungen gesteigert werden kann.

Im darauffolgenden Kapitel werden Analogiebetrachtungen von Optimierungsverfahren und der Produktentwicklung beschrieben. Die Aktivitäten und Prozesse in der Produktentwicklung können auch als kontinuierlicher Optimierungsprozess angesehen werden, wodurch zwischen diesen beiden eine gewisse Konformität besteht. Neben Analogiebetrachtungen evolutionärer Algorithmen, welche im weiteren Verlauf dieser Arbeit bei der Entwicklung der Parallelisierungsmethode sowie in den Fallstudien Anwendung finden, werden auch Optimierungsverfahren vorgestellt, die durch Analogien zum Verhalten interdisziplinärer Entwicklungsteams bei der Produktentwicklung inspiriert sind. Dieser Ansatz wird im weiteren Verlauf der Arbeit bei der Entwicklung der Parallelisierungsmethode aufgegriffen und weiterverfolgt. Weiterhin erfolgt eine Beschreibung des aktuellen Forschungsstands der Autogenetischen Konstruktionstheorie, in der die gesamte Produktentwicklung als evolutionärer Prozess betrachtet wird. Dabei wird herausgestellt, wie insbesondere bei komplexen Evaluationsprozessen durch die entwickelte Parallelisierungsmethode die Effizienz in der Anwendung erhöht werden kann.

Anschließend wird die Effizienzsteigerung von Optimierungsverfahren durch Parallelisierung erläutert. Dies beinhaltet einen Überblick über die theoretischen Grundlagen der Parallelisierung von Optimierungsverfahren durch Modelldekomposition sowie den bisherigen Stand der Technik in der Anwendung. Ausgehend von einer Analyse der in der Produktentwicklung etablierten Methoden zur parallelen Durchführung von Aktivitäten wird die konventionelle Ressourcendefinition der Parallelisierung von Optimierungsverfahren erweitert und unter den Begriffen Concurrent Optimization und Simultaneous Optimization ein neuer Ansatz zur Parallelisierung vorgestellt. Da dieser in der Anwendung eine verteilte IT-Umgebung erfordert, werden die wesentlichen Eigenschaften des verteilten Rechnens erläutert, wobei besonderer Fokus auf der Verwendung von Workstation-Clustern liegt. Zudem werden die Vorteile der prioritätenbasierten Bearbeitung von Berechnungsaufgaben in einer Cluster-Umgebung herausgestellt und ein Überblick der Verfahren zur Ermittlung geeigneter Prozessprioritäten gegeben.

Die zunächst theoretisch entwickelte Methode zur Effizienzsteigerung von Optimierungsverfahren durch Parallelisierung wird anschließend in einem Framework prototypisch umgesetzt. Basierend auf Anforderungen wird das Framework konzeptioniert, wobei die in das Framework integrierten Systeme besonders herausgestellt werden. Die Anforderungen und das entwickelte Konzept bilden die Grundlage für die Integration und Konfiguration des Clusters sowie für die Entwicklung des Optimierungssystems DAG2OPT, welches gleichzeitig für den Anwender das Front-End des Frameworks darstellt. Neben der Definition des Optimierungs- und Evaluationsmodells wird die Optimierung der Prozessprioritäten beschrieben. Dabei werden anhand von drei Testszenarien unterschiedlicher Komplexität verschiedene Scheduling-Verfahren gemäß ihrer Eignung zur Ermittlung der optimalen Prozessprioritäten analysiert und das effektivste Scheduling-Verfahren in das

Optimierungssystem implementiert. Im Anschluss erfolgt die Validierung des Frameworks auf Basis von drei Fallstudien, welche charakteristische multidisziplinäre Optimierungsaufgaben aus realen Entwicklungsprojekten darstellen. Aus den Erkenntnissen dieser Fallstudien werden abschließend Handlungsempfehlungen zur effizienten Durchführung einer Optimierung in dem entwickelten Framework abgeleitet.

2 Technische Grundlagen der Produktoptimierung

In diesem Kapitel werden die technischen Grundlagen der Optimierung von Produkten erläutert. Da in der Literatur keine einheitliche Klassifizierung von Optimierungsmethoden und Optimierungsverfahren vorhanden ist, werden zunächst verschiedene Klassifizierungen aus der Literatur vorgestellt, analysiert und zu einer Gesamtdarstellung synthetisiert. Ausgehend von dieser Klassifizierung erfolgt die Definition der wichtigsten Begriffe.

Weiterhin werden die mathematischen Grundlagen und der generelle Prozess einer Optimierung erläutert. Anschließend erfolgt die Beschreibung des Zusammenhanges zwischen dem Designvariablen- und dem Lösungsraum einer Optimierung. Besonderer Fokus liegt dabei auf der Abbildung des Lösungsraumes aus dem Designvariablenraum und der Rückführung der erzielten Lösungen in den Designvariablenraum zur Bestimmung der optimalen Werte der Designvariablen.

Bei der entwickelten Klassifizierung wird zwischen Optimierungsmethoden und Optimierungsverfahren unterschieden. Daher erfolgt anschließend eine Beschreibung der wichtigsten Methoden und Verfahren. Bei der Beschreibung der Optimierungsverfahren werden die Struktur und die Anwendung genetischer Algorithmen hervorgehoben, da diese besonders zur Lösung diskreter und kombinatorischer Probleme geeignet sind, wie sie auch in dieser Arbeit bei der Optimierung der Prozessprioritäten zu finden sind.

Anschließend erfolgt eine Beschreibung von Verfahren zur explorativen Untersuchung des Lösungsraumes. Diese sind als Ergänzung bzw. Vorbereitung einer Optimierung zu sehen, können aber auch gleichermaßen den hybriden Optimierungsverfahren zugeordnet werden.

Das Kapitel endet mit der Darstellung der wichtigsten Optimierungsstrategien. Dabei werden die wesentlichen Punkte aufgeführt, welche beim Festlegen einer Optimierungsstrategie berücksichtigt werden sollten, um eine Optimierungsaufgabe effektiv und effizient zu bearbeiten. Multikriterielle und multidisziplinäre Optimierungsstrategien sowie Strategien zur Kombination von Optimierungsmethoden und -verfahren werden dabei besonders herausgestellt, da insbesondere durch diese die Effektivität und die Effizienz einer Optimierung entscheidend beeinflusst werden können.

2.1 Klassifizierung, Begriffe und Definitionen

Optimierungsmethoden können nach verschiedenen Kriterien eingeteilt werden, wobei in der Literatur keine einheitliche Klassifizierung vorherrscht. Dies liegt zum einen an der schnellen Entwicklung in diesem Bereich, aber auch an der Vielzahl verschiedener Disziplinen, Verfahren und Algorithmen. Weltweit existieren mehr als 1000 Optimierungsprogramme [Sch13]. Im Folgenden werden verschiedene Klassifizierungen aus der Literatur vorgestellt, analysiert und anschließend zu einer Gesamtdarstellung synthetisiert. Die Reihenfolge der Darstellung ist hierbei nach dem Grad der Komplexität und der Granularität der jeweiligen Klassifizierung gewählt. Darauf aufbauend werden die wichtigsten Begriffe definiert.

Bendsøe und Sigmund [BS04] unterscheiden drei Kategorien in der Ermittlung der optimalen Materialverteilung, um somit das optimale Layout einer Struktur zu ermitteln: die Topologieoptimierung, die Formoptimierung (Shape-Optimierung) und die Dimensionierung (Sizing). Diese drei Kategorien sind in Abbildung 2.1 dargestellt.

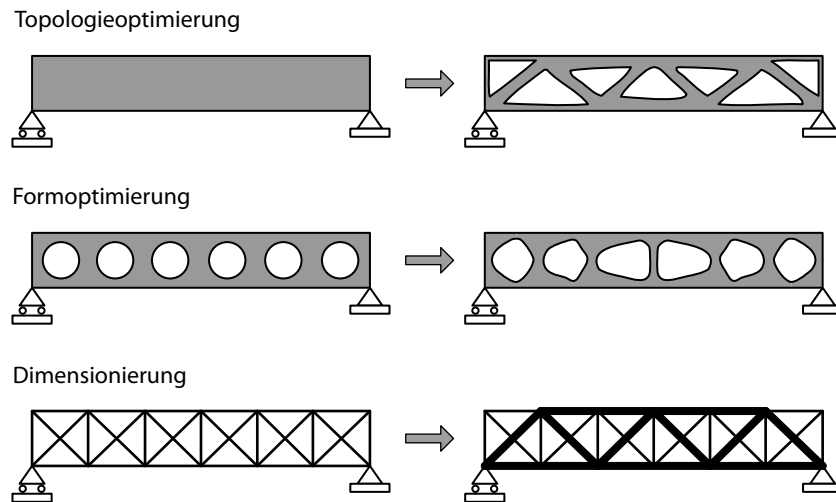


Abbildung 2.1: Kategorien der Strukturoptimierung, nach [BS04]

Jede der drei Kategorien spricht einen anderen Aspekt des zu optimierenden Bauteils an. Die Topologie eines Bauteils beschreibt die reine Anordnung der das Bauteil definierenden Objekte. So wird durch die Optimierung der Lage und der Anordnung von Strukturelementen wie z. B. Rippen, Stegen, Bohrungen und Hohlräumen die optimale Materialverteilung des Bauteils ermittelt. Ziel der Formoptimierung ist es, die optimale Form der begrenzenden Flächen der Struktur zu ermitteln. Bei der Dimensionierung werden die optimalen Wandstärken eines Bauteils bestimmt. Hierbei müssen die Designvariablen der zu optimierenden Struktur bereits im Vorfeld bekannt sein und können sich im Laufe der Optimierung nicht ändern [BS04].

Diese Einteilung geht lediglich auf die unterschiedliche Verteilung des Materials ein. Die Art der Designvariablen oder die Verwendung verschiedener Materialien werden nicht berücksichtigt.

Harzheim [Har14] verwendet die gleiche Unterteilung. Innerhalb der Formoptimierung wird zwischen CAD-basierter und FE-Netz-basierter Formoptimierung unterschieden. Bei der CAD-basierten Optimierung sind die Designvariablen direkt mit den Parametern eines parametrischen CAD-Modells verknüpft. Es wird also zunächst das CAD-Modell verändert und anschließend vernetzt und berechnet. Dem gegenüber steht die FE-Netz-basierte Formoptimierung, bei der die Modellbeschreibung direkt über die finiten Elemente (FE) erfolgt. Die Designvariablen werden hierbei durch die Elementknoten des FE-Netzes repräsentiert [Har14]. Insbesondere die Unterteilung nach der Definition der Designvariablen führt zu einer grundlegenden Unterscheidung und muss bei der Bearbeitung einer Optimierungsaufgabe bereits frühzeitig berücksichtigt werden.

Innerhalb der Topologieoptimierung erfolgt die Unterscheidung zwischen der mathematischen Topologieoptimierung, bei der ein mathematisches Optimierungsproblem gelöst wird, und der empirischen Topologieoptimierung, welche empirische Iterationsvorschrif-

ten verwendet [Har14]. Beide Methoden basieren jedoch auf einem FE-Netz, wodurch keine Unterscheidung nach der Definition der Designvariablen stattfindet.

Ramm et al. [RMS98] stellen eine erweiterte Unterscheidung vor. Unter Berücksichtigung der verwendeten Designvariablen werden vier Disziplinen der Strukturoptimierung unterschieden: die Topologieoptimierung, die Formoptimierung, die Dimensionierung und die Materialoptimierung. Aufgrund ihres unterschiedlichen Komplexitätsgrades und der Art der Ergebnisse werden die Disziplinen zudem als Stufen in einer zeitlichen Abfolge unterschieden. Durch die Topologieoptimierung wird ein erster Entwurf für das grundsätzliche Layout eines Bauteils generiert. Auf dessen Basis werden anschließend mittels der Formoptimierung die Begrenzungsflächen des Bauteils optimiert, um die mechanischen Anforderungen zu erfüllen. An letzter Stelle erfolgt die Dimensionierung und die Optimierung des Materials [RMS98]. Die Disziplinen nach Ramm et al. sind in Abbildung 2.2 dargestellt. Im Gegensatz zur Darstellung in der Abbildung wird bei der Dimensionierung nicht zwangsläufig eine Verkleinerung des Querschnittes erzielt. Wird die Optimierung z. B. mit Spannungswerten oberhalb der zulässigen Spannungsrestriktion begonnen, sind auch Querschnittsvergrößerungen möglich.

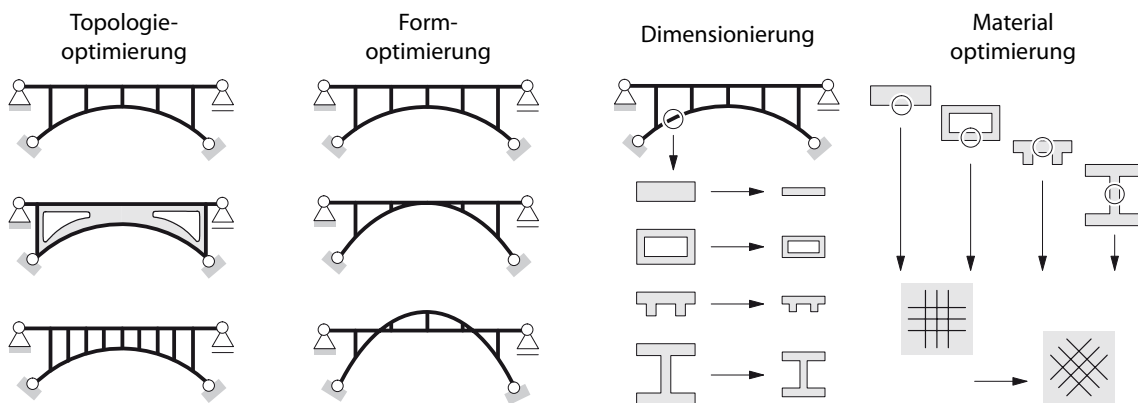


Abbildung 2.2: Klassifizierung von Optimierungsdisziplinen, nach [RMS98]

Schumacher [Sch13] klassifiziert Strukturoptimierungsaufgaben nach der Art der Designvariablen, da sich nach diesen auch die anzuwendende Lösungsstrategie richtet [SM63]. Es werden folgenden Arten unterschieden:

- Topologieoptimierung
- Formoptimierung
- Dimensionierung
- Wahl der Bauweise
- Wahl der Materialeigenschaften

Die Darstellung nach Ramm et al. wird also um die Wahl der Bauweise erweitert. Bei der Wahl der Bauweise werden unterschiedliche Varianten miteinander verglichen, zwischen denen kein kontinuierlicher Übergang möglich ist, z. B. die Querschnittsarten eines Trägers.

Die Topologieoptimierung kann zusätzlich nach der Definition des Topologieräume, der Art der Ziel- und Restriktionsfunktionen, der Definition der Designvariablen und dem

verwendetem Lösungsalgorithmus unterteilt werden [Sch13]. Der Topologieraum beinhaltet den zur Verfügung stehenden Bauraum, aber auch die Definition der Strukturobjekte. Diese können als diskrete oder kontinuierliche Strukturen definiert werden. Auch hier stellt die Unterscheidung nach der Definition der Designvariablen eine grundlegende Unterteilung dar, welche bereits frühzeitig bei der Bearbeitung einer Optimierungsaufgabe berücksichtigt werden muss. Daher wird diese Einteilung bei der Synthese der Klassifizierungen ebenfalls frühzeitig berücksichtigt. Die weiteren Klassifizierungen greifen nicht auf der bisher vorgestellten Ebene der Optimierungsdisziplinen, sondern setzen ein bereits parametrisiertes Optimierungsproblem voraus. Daher werden die Disziplinen Topologieoptimierung und Formoptimierung nicht erwähnt.

Baier et al. [BSS94] geben einen Überblick über gängige Optimierungsverfahren unterscheiden zwischen der Lösungsstrategie und dem Lösungs- bzw. Suchrichtungserzeuger. Die Lösungsstrategie beschreibt die prinzipielle Vorgehensweise zur Lösung eines Optimierungsproblems (z. B. die Linearisierung der Zielfunktion oder die Implementierung von Restriktionen). Der Lösungserzeuger übernimmt innerhalb der gewählten Strategie die iterative Veränderung der Designvariablen und repräsentiert somit das Optimierungsverfahren [BSS94]. Dieses Gliederungsschema ist in Abbildung 2.3 dargestellt.

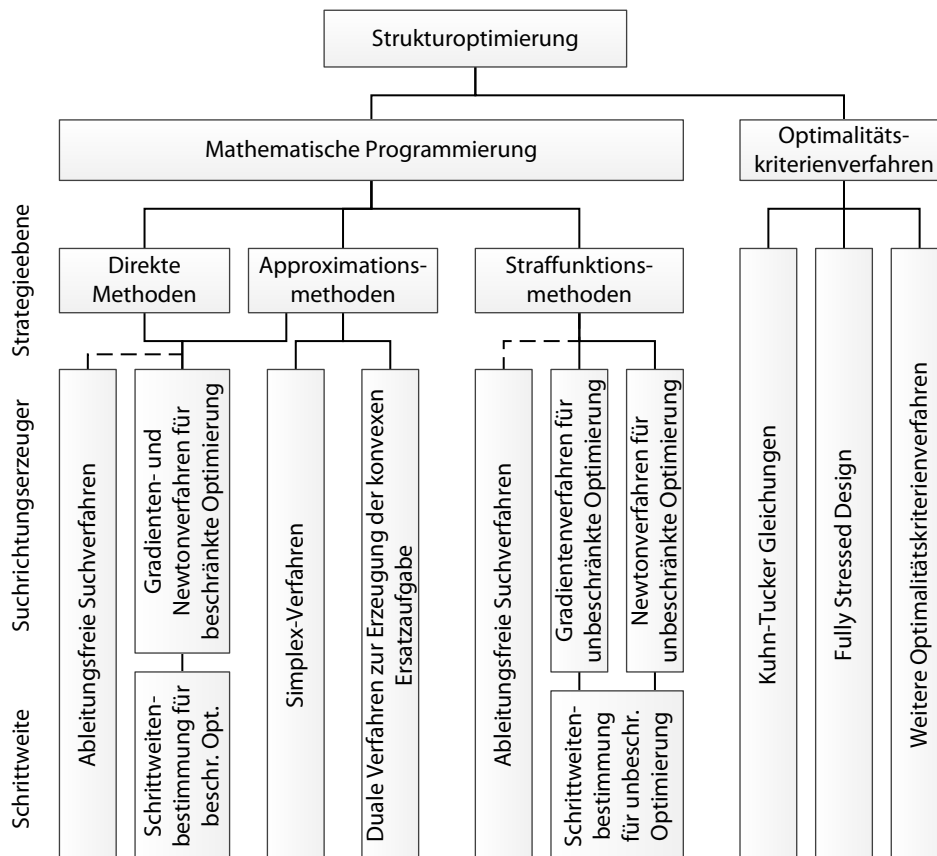


Abbildung 2.3: Klassifizierung nach Strategieebene, Suchrichtungserzeuger und Schrittweite, nach [BSS94]

Das vorgestellte Schema gibt einen Überblick über verschiedene Gliederungsebenen hinweg. Auf oberster Ebene wird zwischen der mathematischen Programmierung und den Optimalitätskriterienverfahren unterschieden. Die ableitungsfreien Suchverfahren können

nicht eindeutig einer bestimmten Strategie zugeordnet werden. Daher ist diese Verbindung gestrichelt dargestellt. Eine weitere Einteilung nach der Art der Parametrisierung oder der zu erwartenden Ergebnisse wird nicht gegeben. Auf der gleichen Abstraktionsebene kann zusätzlich unter Berücksichtigung der Parametrisierung zwischen der parametrischen Optimierung und der parameterfreien Optimierung mit Optimalitätskriterien unterschieden werden [Maj14]. Für eine detaillierte Beschreibung der Optimalitätskriterienverfahren wird auf [BSS94] verwiesen.

Cavazzuti [Cav13] unterteilt Optimierungen in drei Kategorien: die statistische Versuchsplanung (Design of Experiments, DoE), Optimierungsalgorithmen und Robustheitsanalysen. Die statistische Versuchsplanung beinhaltet die Generierung von antwortflächenbasierten Metamodellen (Response Surface Model, RSM). Es wird zwischen interpolierenden und approximierenden RSM unterschieden. Bei den Optimierungsalgorithmen wird zwischen deterministischer und stochastischer Optimierung unterschieden. Innerhalb der deterministischen Optimierung kann die Optimierung ohne oder mit Restriktionen erfolgen. Die stochastische Optimierung beinhaltet evolutionäre Algorithmen und sonstige Algorithmen sowie die monokriterielle und die multikriterielle Optimierung [Cav13].

Die dritte Gruppe bilden die Robustheitsanalysen. Diese beinhalten multikriterielle Robustheitsoptimierungen, bei denen während der Optimierung die Streuung der Designvariablenwerte berücksichtigt wird, und Zuverlässigkeitsanalysen, in denen die Robustheit des Optimierungsergebnisses bezogen auf die Streuung der Designvariablenwerte überprüft wird. Robustheitsanalysen werden in der vorliegenden Arbeit nicht näher erläutert. Für weiterführende Informationen wird auf [Cav13] verwiesen. Die Unterteilung der Optimierungsmethoden nach Cavazzuti ist in Abbildung 2.4 dargestellt.

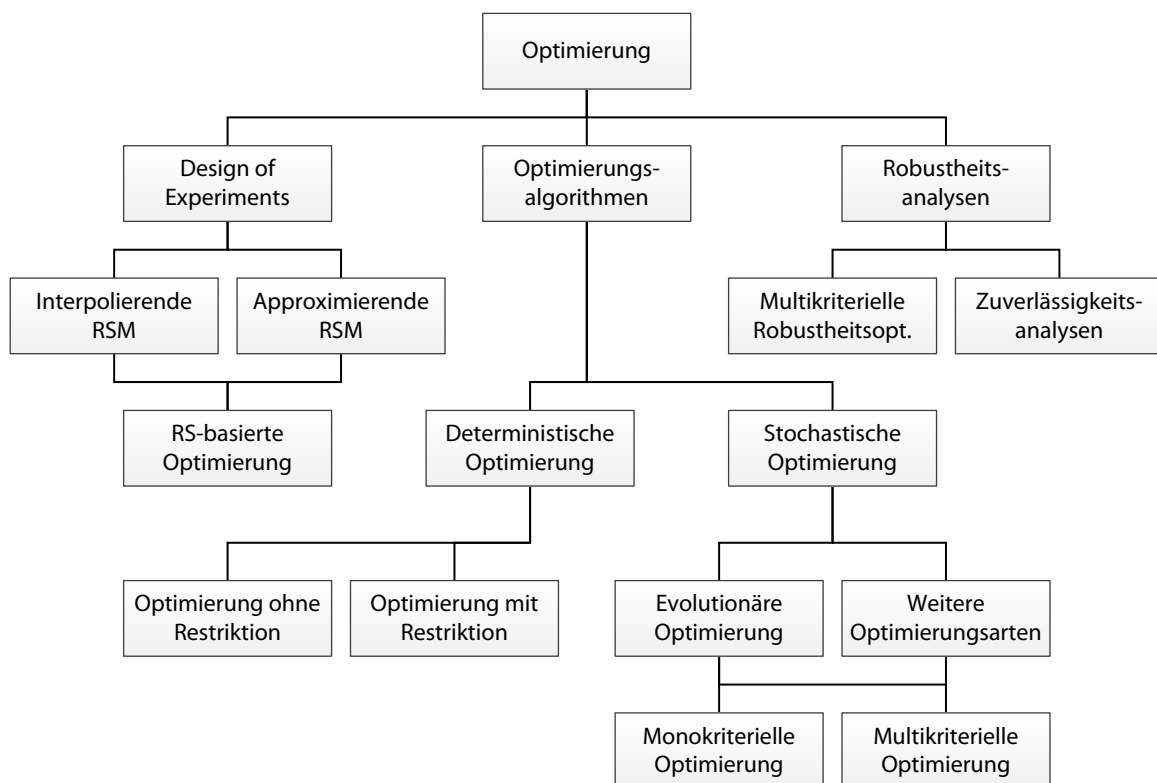


Abbildung 2.4: Hierarchische Unterteilung von Optimierungsmethoden, nach [Cav13]

Die Klassifizierungen nach Baier et al. und Cavazzuti geben bereits einen Überblick über die große Anzahl unterschiedlicher Optimierungsmethoden und -verfahren sowie die verschiedenen Unterteilungsmöglichkeiten. Zudem zeigt sich in den schematischen Darstellungen, dass die Formoptimierung auf unterschiedliche Art und Weise interpretiert werden und somit auch verschiedenartige Lösungen generieren kann (s. Abbildung 2.1 und 2.2).

Auch die Topologieoptimierung kann nicht eindeutig in die dargestellten Klassifizierungen eingeordnet werden. Ein Großteil der Topologieoptimierungsmethoden basiert rein auf dem FE-Modell [SM13], was zu einer großen Anzahl an Designvariablen führt. Der Anwender definiert dabei lediglich den zur Verfügung stehenden Bauraum, welcher auch als Topologieraum bezeichnet wird und aus vielen kleinen Strukturbereichen besteht, die jeweils durch mindestens eine Designvariable beschrieben werden. Aufgrund der Modellrepräsentation durch diese vielen kleinen Bereiche wird die Methode auch als *Pixel*-Methode bezeichnet. Im einfachsten Fall wird jeder dieser Strukturbereiche durch ein finites Element repräsentiert. Die optimale Topologie wird dann durch die Variation des Materialverhaltens der einzelnen Bereiche ermittelt [Sch13]. Da der Anwender in der Regel lediglich den Bauraum definiert und nicht jeden Bereich separat parametrisiert, erfolgt die Definition der Designvariablen durch das System und somit implizit.

Eine weitere Form der Topologieoptimierung ist die Methode der parametrisierten Randbeschreibung (*Bubble*-Methode) [EKS94], welche auf dem CAD-Modell eines Bauteils basiert und eine Kombination aus Topologieoptimierung und Formoptimierung darstellt. Dabei werden in das Bauteil nach einer ersten Formoptimierung iterativ Löcher eingebracht, die dann wiederum einer Formoptimierung unterzogen werden. Dieser Prozess läuft solange, bis die optimale Topologie ermittelt wurde. Aufgrund der freieren Modellbeschreibung des CAD-Modells ist diese Methode wesentlich flexibler, da auch unterschiedliche Evaluationsmodelle verwendet werden können. Nachteilig ist der hohe Modellierungsaufwand für das parametrische CAD-Modell zu sehen [Sch13]. Anhand der beschriebenen Formen der Topologieoptimierung ist zu erkennen, dass zwar jeweils die Topologie eines Bauteils optimiert wird, die Definition der Designvariablen sich jedoch grundlegend voneinander unterscheidet.

Analog zur Topologieoptimierung können auch die Formoptimierung und die Dimensionierung in der Definition der Designvariablen unterschieden werden. Basiert das Evaluationsmodell der Optimierung rein auf einem FE-Modell, ist dies in den meisten Fällen ebenfalls gleichbedeutend mit sehr vielen Designvariablen, welche durch den Anwender aufgrund ihrer hohen Anzahl nicht explizit definiert werden können, sondern vom Optimierungssystem implizit definiert werden. Bei der Formoptimierung können dabei z. B. die Koordinaten der FE-Knoten, bei der Dimensionierung z. B. die Elementdicken von Schalenelementen die Designvariablen darstellen. Daher werden für diese Methoden auch die Begriffe *Free Shape-Optimierung* oder *Free Size-Optimierung* verwendet [Alt15]. Wird hingegen ein parametrisches CAD-Modell verwendet, werden die Designvariablen meist explizit vom Anwender definiert.

Keine der beschriebenen Klassifizierungen erlaubt die eindeutige Zuordnung aller Optimierungsmethoden und -verfahren. Daher wird im Folgenden aus den beschriebenen Klassifizierungen eine eigene Klassifizierung synthetisiert.

Die Definition der Designvariablen stellt eine grundlegende Entscheidung bei der Bearbeitung einer Optimierungsaufgabe dar und muss bereits frühzeitig bedacht werden. Danach richtet sich auch die Wahl der Optimierungsmethode und die Art der zu erwartenden

Lösung. Die explizite Definition der Designvariablen bildet hierbei den komplexesten aber auch den flexibelsten Ansatz [RMS98]. Eine genaue Klassifizierung der Optimierungsdisziplinen, -methoden, -verfahren und -algorithmen ist entscheidend für die Festlegung einer Optimierungsstrategie und somit die effiziente Lösung einer Optimierungsaufgabe. Die aus den vorangegangenen Überlegungen entstandene Klassifizierung ist in Abbildung 2.5 dargestellt.

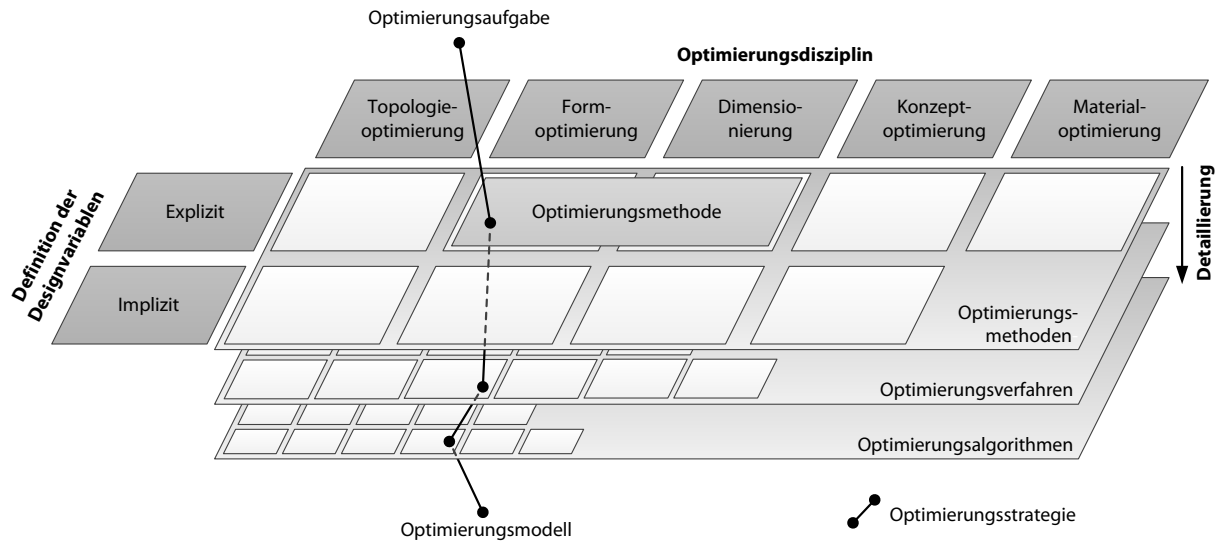


Abbildung 2.5: Klassifizierung von Optimierungsmethoden und Vorgehen zur Festlegung der Optimierungsstrategie

Die von Schumacher [Sch13] verwendeten Optimierungsdisziplinen werden im Wesentlichen beibehalten. Jedoch wird die Wahl der Bauweise durch die Konzeptoptimierung ersetzt, da diese umfassender in der Beschreibung ist. Ein Konzept stellt einen Entwurfsvorschlag zur Lösung eines Problems dar, welcher ausreichend detailliert ist, um die Eignung des Entwurfs zur Lösung des Problems bewerten zu können [AHC15]. Die Konzeption beinhaltet also die Erstellung und Auswahl von Prinziplösungen einer Entwicklungsaufgabe [VDI93]. Da dieser Prozess auf verschiedenen Stufen der Abstraktion und Detaillierung erfolgen kann, umfasst die Konzeption somit auch die Wahl der Bauweise eines Systems oder einer Komponente [See05]. Kennzeichnend für die Konzeptoptimierung ist die Auswahl aus verschiedenen Varianten, zwischen denen kein kontinuierlicher Übergang existiert. Konzeptoptimierungen repräsentieren somit diskrete, kombinatorische Optimierungsaufgaben. Eine beispielhafte Anwendung der Konzeptoptimierung wird in Abschnitt 3.2.3 beschrieben.

Neben der Klassifizierung der Optimierungsdisziplinen findet eine weitere Klassifizierung auf Basis der Definition der Designvariablen statt. In der daraus resultierenden Matrixdarstellung können alle Optimierungsmethoden eindeutig zugeordnet werden. Eine Optimierungsmethode kann hierbei auch mehrere Optimierungsdisziplinen abdecken. So können z. B. durch explizit definierte Designvariablen in einem parametrischen CAD-Modell je nach Parametrisierung und Art der verwendeten Parameter sämtliche Optimierungsdisziplinen bedient werden (s. Abschnitt 2.5.1). Methoden mit impliziter Designvariablen-Definition stellen aufgrund der einfachen Anwendung durch den Nutzer oft Speziallösungen bezogen auf die jeweilige Optimierungsdisziplin dar.

Die dargestellte Klassifizierung stellt zudem ein Vorgehensmodell zur Festlegung einer Optimierungsstrategie dar. Ausgehend von einer Optimierungsaufgabe wird eine Optimierungsmethode ausgewählt, welche die Grundlage für die Auswahl eines Optimierungsverfahrens und schließlich eines konkreten Optimierungsalgorithmus bildet. Da ein Optimierungsalgorithmus die konkrete Umsetzung eines Optimierungsverfahrens ist, erfolgt somit eine Detaillierung in der Vorgehensweise zur Bearbeitung der Optimierungsaufgabe. Dies beinhaltet zudem die Festlegung der Designvariablen sowie die Definition von Ziel- und Restriktionsfunktionen. Das Ergebnis dieses Prozesses stellt das Optimierungsmodell dar.

Aus der hergeleiteten Klassifizierung werden die wesentlichen Begriffe extrahiert und für den weiteren Verlauf dieser Arbeit definiert. Die Reihenfolge der Begriffe entspricht dem jeweiligen Detaillierungsgrad gemäß Abbildung 2.5.

Optimierungsstrategie: Die Optimierungsstrategie legt das Vorgehen zum Lösen einer Optimierungsaufgabe fest. Sie stellt also den gewählten Lösungsansatz dar. Dies beinhaltet die Festlegung der Designvariablen, der Zielfunktion und der Restriktionen. Weiterhin werden je nach zu lösender Aufgabe die Optimierungsmethode, das Optimierungsverfahren und der passende Optimierungsalgorithmus ausgewählt. Eine Optimierungsstrategie kann auch die Kombination verschiedener Optimierungsmethoden und -verfahren beinhalten, z. B. in sequentieller oder paralleler Abfolge oder in einer Multi-Level-Optimierung.

Optimierungsdisziplin: Optimierungsaufgaben können in verschiedene Optimierungsdisziplinen eingeteilt werden. Die Einteilung erfolgt nach der Art der Funktionsweise, der Komplexität, der Charakteristik der Designvariablen und nach der Art der zu erwartenden Lösung. Optimierungsdisziplinen sind: Topologieoptimierung, Formoptimierung, Dimensionierung, Konzeptoptimierung und Materialoptimierung.

Optimierungsmethode: Die Optimierungsmethode stellt die zum Lösen einer Optimierungsaufgabe verwendete Funktionsweise dar. Dabei werden eine oder mehrere Optimierungsdisziplinen verwendet. Weiterhin unterscheiden sich Optimierungsmethoden in der Art der Parameterdefinition bei der Festlegung der Designvariablen. Hierbei können die Parameter der Optimierung vom Anwender explizit definiert oder vom System definiert werden (implizit). Zudem ist eine Kombination beider Parameterdefinitionen möglich. In einer Optimierungsmethode können verschiedene Verfahren oder Algorithmen verwendet werden.

Optimierungsverfahren: Die zur Lösung einer Optimierungsaufgabe verwendeten Ansätze und Optimierungsalgorithmen werden zusammengefasst als Optimierungsverfahren bezeichnet [Sch13].

Optimierungsalgorithmus: Ein prozeduraler Algorithmus ist eine genau beschriebene Vorschrift zur Lösung einer Klasse von Problemen mittels einer endlichen Folge von eindeutig bestimmten und tatsächlich durchführbaren Teilhandlungen [EBS15], [FH11]. Dabei wird eine Eingabe oder eine Menge von Eingaben zu einer Ausgabe oder einer Menge von Ausgaben verarbeitet [LRL03]. Ein Optimierungsalgorithmus stellt die konkrete Umsetzung eines Optimierungsverfahrens dar. Er minimiert oder maximiert eine gegebene Zielfunktion unter oder ohne Berücksichtigung von Restriktionen [Sch13].

Optimierungsmodell: Das Optimierungsmodell beinhaltet den Optimierungsalgorithmus, die Definition und Generierung der Designvariablen sowie die Verarbeitung der Zielfunktion und der Restriktionen.

Evaluationsmodell: Das Evaluationsmodell enthält die für die Auswertung einer vom Optimierungsalgorithmus generierten Lösung notwendige mathematische Beschreibung. Diese kann in analytischer Form oder numerischer Form vorliegen. Das Evaluationsmodell beinhaltet somit die Verarbeitung der Designvariablen sowie die Bestimmung des Zielfunktionswertes und der Restriktionen [EKS94].

Je nach Komplexität der Optimierungsaufgabe und der gewählten Optimierungsstrategie kann innerhalb des Evaluationsmodells zusätzlich zwischen Entwurfsmodell und Analysemodell unterschieden werden:

- Das *Entwurfsmodell* bildet die Verbindung zwischen der abstrakten Parameterformulierung des Optimierungsmodells und der notwendigen Repräsentation eines Bauteils für die Analyse. Es kann also z. B. der Beschreibung der Geometrie und dem materiellen Aufbau eines Bauteils dienen.
- Das *Analysemodell* stellt die mathematische Beschreibung für die Analyse der vom Entwurfsmodell bereitgestellten Daten dar. Dies beinhaltet z. B. die Berechnung der Strukturantwort eines Bauteils sowie die Durchführung der Sensitivitätsanalyse [Sch01b].

Durch die Modelltrennung kann eine sehr flexible modulare Umgebung für die Durchführung einer Optimierung erzeugt werden. Es können sehr schnell z. B. die Zielfunktion und der Optimierungsalgorithmus im Optimierungsmodell, die Geometrieerzeugung im Entwurfsmodell oder die Analysemethode im Analysemodell unabhängig voneinander verändert werden [RMS98]. Auch wenn Eschenauer eine etwas andere Einteilung und Begriffsdefinition verwendet, so führt auch er die gleichen Vorteile der Modelltrennung in seinem Drei-Säulen-Modell auf [Esc85].

Parameter: Die Eigenschaften eines Systems bzw. Modells werden durch Parameter und Zustandsvariablen erfasst. Parameter repräsentieren dabei die definierenden Eigenschaften des Systems, welche nicht zwangsläufig konstant sein müssen, sondern in Parameterstudien oder Optimierungen variiert werden können [VWBZ09]. Sie werden dann durch den Anwender explizit als Designvariablen definiert.

Designvariablen: Die Designvariablen sind die bei der Optimierung eines Systems bzw. Modells vom Optimierungsalgorithmus veränderbaren Parameter. Diese können vom Anwender explizit oder implizit vom System definiert sein. Beispiele für explizit definierte Designvariablen sind Wandstärke, Steuerungsparameter von Skizzen und Splines oder Materialkennwerte. Implizit definierte Designvariablen sind z. B. die Dichtewerte der finiten Elemente bei der Topologieoptimierung.

Designvariablenraum: Der Designvariablenraum ist der Bereich, in dem die Designvariablen vom Optimierungsalgorithmus verändert werden können. In der Regel wird der Designvariablenraum durch die unteren und oberen Grenzen der Designvariablen festgelegt [Sch13].

Lösungsraum: Der Lösungsraum stellt das Gegenstück des Designvariablenraumes auf der Ergebnisseite der Zielfunktionsformulierung dar. Jedem Punkt im Designvariablenraum kann ein Punkt im Lösungsraum zugeordnet werden.

Zustandsvariablen: Zustandsvariablen repräsentieren die beschreibenden Eigenschaften des Systems bzw. Modells. Bei einer Optimierung werden die relevanten Ausgabewerte des Analysemodells durch die Zustandsvariablen dargestellt. Sie bilden die Ergebnisse

einer Simulation ab und werden für die Berechnung des Zielfunktionswertes oder zur Ermittlung der Restriktionswerte benötigt. Jedem Vektor aus Designvariablen des Designvariablenraumes steht ein Vektor aus Zustandsvariablen und somit eine definierte Lösung im Lösungsraum gegenüber. Zustandsvariablen, welche in einer Zielfunktion verwendet werden, werden auch als Zielkriterien bezeichnet.

Effektivität: Die Effektivität einer Optimierung beschreibt die Qualität der erzielten Lösung bzw. des ermittelten Optimums. Eine hohe Effektivität liegt vor, wenn das globale Optimum einer Optimierungsaufgabe ermittelt wird oder eine hohe Wahrscheinlichkeit besteht, das globale Optimum zu finden.

Effizienz: Die Effizienz einer Optimierung gibt an, wie schnell die Lösung der Optimierungsaufgabe erreicht wird. Dabei werden die Anzahl der benötigten Evaluationen und der mögliche Parallelisierungsgrad bezogen auf die zur Verfügung stehenden Ressourcen berücksichtigt.

Somit sind die wichtigsten Begriffe definiert. Für zusätzliche Begriffsdefinitionen wird auf weiterführende Literatur verwiesen, z. B. [Esc85], [Sch13], [Cav13], [Har14].

2.2 Mathematische Formulierung einer Optimierungsaufgabe

Um eine Optimierungsaufgabe zu lösen, muss diese zunächst mathematisch formuliert werden. Die mathematische Formulierung stellt eine abstrakte Definition der Aufgabe dar. Die Optimierungsaufgabe besteht im Allgemeinen darin, die Werte der Designvariablen, welche durch den Vektor \mathbf{x} der Länge n beschrieben werden, so zu verändern, dass die von den Designvariablen abhängige Zielfunktion $f(\mathbf{x})$ minimiert wird [Sch01b], [Sch13].

$$\min f(\mathbf{x}) \tag{2.1}$$

Je nach Problemstellung sind bei der Optimierung Restriktionen bzw. Nebenbedingungen einzuhalten. Hierbei wird zwischen Ungleichheitsrestriktionen (2.2) und Gleichheitsrestriktionen (2.3) unterschieden, durch die der Lösungsraum der Optimierung eingeschränkt wird.

$$g_j(\mathbf{x}) \leq 0 \quad \text{mit } j = 1, n_g \tag{2.2}$$

$$h_k(\mathbf{x}) = 0 \quad \text{mit } k = 1, n_h \tag{2.3}$$

Existieren mehrere Ungleichheitsrestriktionen $g_j(\mathbf{x})$ bzw. Gleichheitsrestriktionen $h_k(\mathbf{x})$, können diese auch als Ungleichheitsrestriktionsvektor $\mathbf{g}(\mathbf{x})$ bzw. Gleichheitsrestriktionsvektor und $\mathbf{h}(\mathbf{x})$ zusammengefasst werden.

Eine weitere Einschränkung erfolgt durch die oberen und unteren Grenzen der Designvariablen x_i^l und x_i^u (engl. lower and upper boundaries). Durch diese expliziten Restriktionen wird zunächst der Designvariablenraum und als Konsequenz daraus der Lösungsraum eingeschränkt.

$$\mathbf{x} = \begin{bmatrix} x_1 \\ \vdots \\ x_n \end{bmatrix} \quad \text{mit } x_i^l \leq x_i \leq x_i^u; \quad i = 1, n \tag{2.4}$$

Zusammenfassend kann die mathematische Beschreibung einer Optimierungsaufgabe auch wie folgt formuliert werden:

$$f(\mathbf{x}^*) = \min_x \{f(\mathbf{x}) \mid \mathbf{x} \in X\} \quad \text{mit} \quad X = \{\mathbf{x} \in \mathbb{R}^n \mid \mathbf{g}(\mathbf{x}) \leq \mathbf{0}, \mathbf{h}(\mathbf{x}) = \mathbf{0}\} \quad (2.5)$$

Der Term $f(\mathbf{x}^*)$ stellt hier den Zielfunktionswert der optimierten Designvariablen \mathbf{x}^* dar und somit das gefundene Optimum. Weiterhin repräsentiert X den zulässigen Designvariablenraum, \mathbb{R}^n die n -dimensionale Menge der reellen Zahlen, $\mathbf{g}(\mathbf{x})$ den Ungleichheitsrestriktionsvektor und $\mathbf{h}(\mathbf{x})$ den Gleichheitsrestriktionsvektor [Sch13].

Besteht die Optimierungsaufgabe darin, ein oder mehrere Zielkriterien zu maximieren, werden diese mit dem Faktor -1 multipliziert. Die Beschränkung auf die Minimierung der Zielfunktion stellt somit kein Problem dar und Minimierungs- und Maximierungsaufgaben können gleichwertig betrachtet werden. Ungleichheitsrestriktionen und Gleichheitsrestriktionen sind auf die gleiche Weise transformierbar [Sch13], [Har14].

Die Lösung einer Optimierungsaufgabe stellt nie ein allgemeingültiges Optimum dar, sondern stets das Optimum im Bezug auf die geltenden Ausgangsbedingungen, die Zielfunktionsformulierung und die Restriktionen. Das Ergebnis ist also stets relativ zu seiner Umgebung zu sehen.

In den meisten Fällen weist die zu optimierende Zielfunktion mehrere lokale Minima auf, von denen nur eines das globale bzw. absolute Minimum darstellt. Viele Optimierungsalgorithmen finden nur das nächstgelegene lokale Minimum. Insbesondere Algorithmen, welche eine Suchrichtung verwenden, bewegen sich von dem Startpunkt aus immer abwärts zum nächstgelegenen Minimum [Har14]. Der Unterschied zwischen lokalem und globalem Minimum ist in Abbildung 2.6 dargestellt.

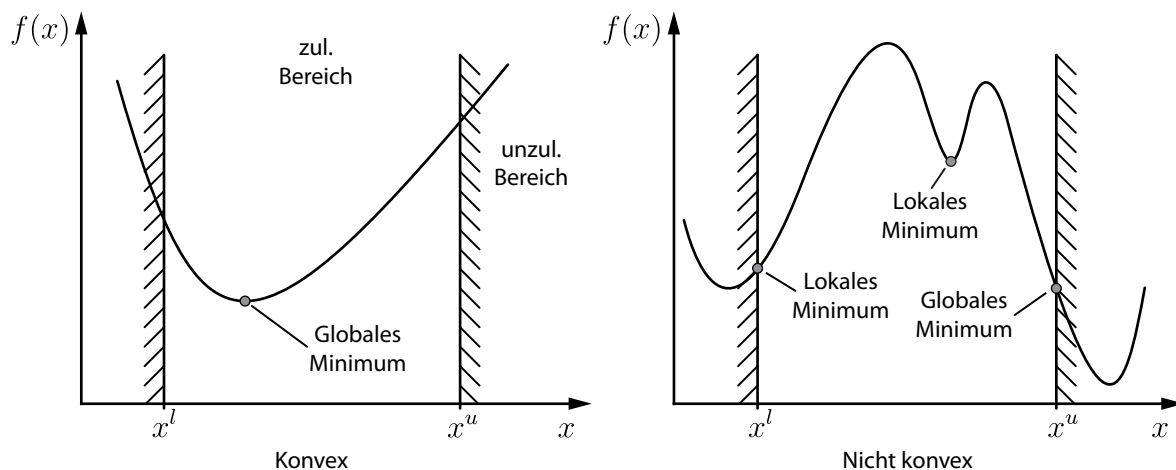


Abbildung 2.6: Lokales und globales Minimum eindimensionaler Zielfunktionen, nach [Har14]

Soll in diesem Fall sichergestellt werden, dass das globale Minimum gefunden wird, so darf die Zielfunktion nur ein Minimum aufweisen, was durch die Konvexität der Funktion sichergestellt wird. Eine Funktion $f(\mathbf{x})$ mit $\mathbf{x} \in [\mathbf{x}^l, \mathbf{x}^u]$ heißt konvex, wenn gilt:

$$f(\Theta \cdot \mathbf{x}_a + (1-\Theta) \cdot \mathbf{x}_b) \leq \Theta \cdot f(\mathbf{x}_a) + (1-\Theta) \cdot f(\mathbf{x}_b) \quad \text{mit} \quad \mathbf{x}_a, \mathbf{x}_b \in [\mathbf{x}^l, \mathbf{x}^u]; \quad \Theta \in [0, 1] \quad (2.6)$$

Die Funktionswerte $f(\mathbf{x})$ (linke Seite der Ungleichung) müssen somit auf oder unterhalb der Geraden zwischen den Punkten \mathbf{x}_a und \mathbf{x}_b sein (rechte Seite der Ungleichung). Diese Bedingung muss für jede beliebige Wahl der Punkte \mathbf{x}_a und \mathbf{x}_b im Definitionsbereich zwischen den Grenzen \mathbf{x}^l und \mathbf{x}^u gewährleistet sein. Der Term Θ repräsentiert eine Laufvariable, über welche die zu überprüfenden Punkte auf der Funktion und der Geraden abgebildet werden. Weiterhin ist die Bedingung der Konvexität nur hinreichend und nicht notwendig. Eine nicht konvexe Funktion muss somit nicht zwangsweise lokale Minima aufweisen [Har14].

Die Bedingung der Konvexität für einen eindimensionalen Designvariablenraum ist grafisch in Abbildung 2.7 dargestellt.

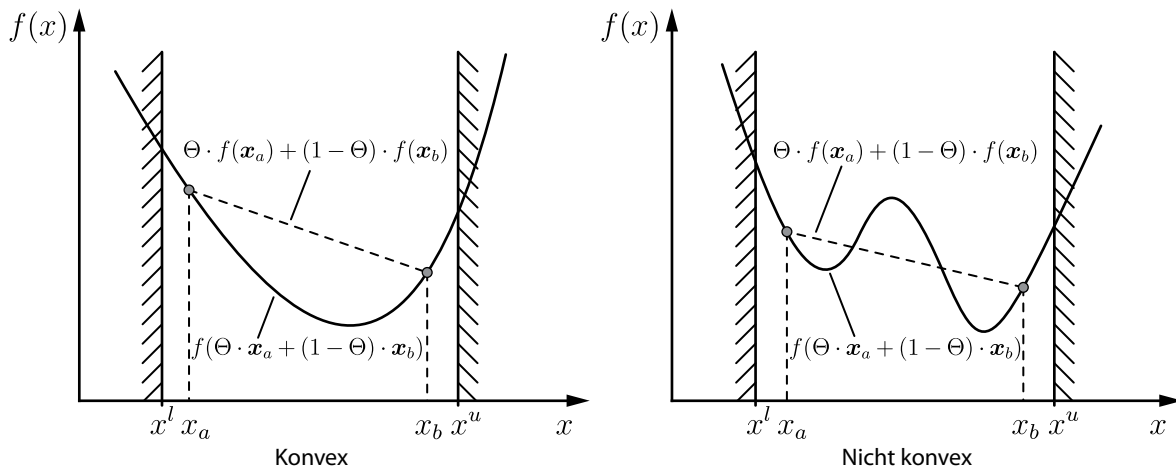


Abbildung 2.7: Definition der Konvexität einer Funktion, nach [Har14]

Werden beim Bearbeiten einer Optimierungsaufgabe Restriktionen berücksichtigt, ist die reine Konvexität der Zielfunktion nicht ausreichend, um das Vorhandensein nur eines globalen Optimums sicherzustellen. Wie in Abbildung 2.8 zu sehen, werden durch die Berücksichtigung einer nicht konvexen Restriktion ein lokales und ein globales Minimum erzeugt, wodurch auch der zulässige Bereich im Designvariablenraum nicht konvex ist.

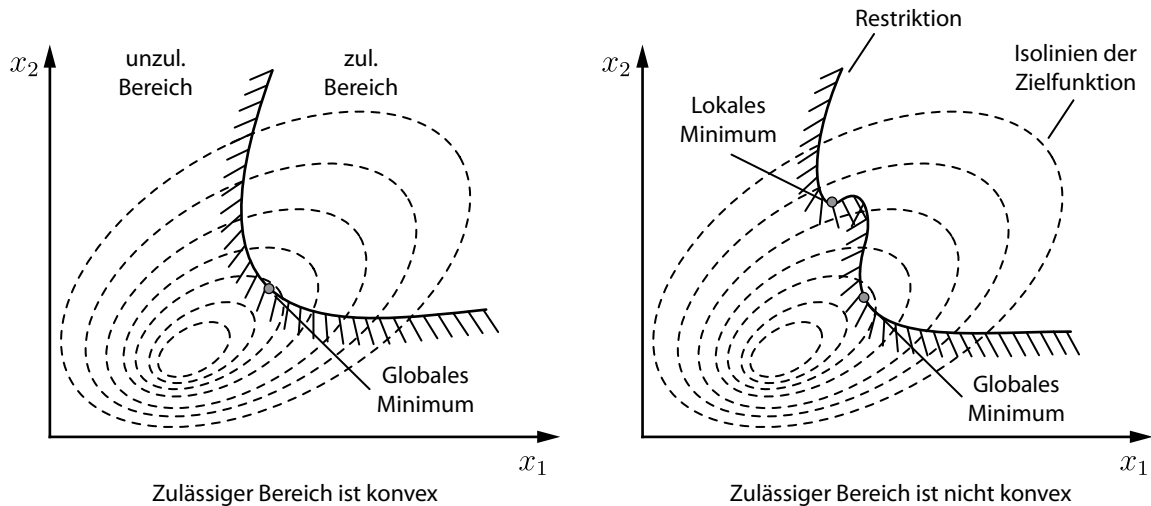


Abbildung 2.8: Konvexität des zulässigen Bereichs einer Funktion, nach [Har14]

Der aus der Zielfunktion resultierende Lösungsraum ist durch die Isolinien der Zielfunktion abgebildet. Alle Punkte auf einer Isolinie liefern hierbei den gleichen Zielfunktionswert. Sie sind also als gleichwertig anzusehen, sind jedoch nicht gleichartig.

Da der zulässige Wertebereich durch eine Menge repräsentiert wird, muss zusätzlich zur Konvexität der Zielfunktion auch die Definition der Konvexität einer Menge berücksichtigt werden. Eine Menge M heißt konvex, wenn gilt:

$$\mathbf{y} = \Theta \cdot \mathbf{x}_a + (1 - \Theta) \cdot \mathbf{x}_b \in M \quad \text{mit} \quad \mathbf{x}_a, \mathbf{x}_b \in M ; \quad \Theta \in [0, 1] \quad (2.7)$$

Durch diese Bedingung wird sichergestellt, dass alle Konvexkombinationen, also die Punkte auf der Strecke zwischen \mathbf{x}_a und \mathbf{x}_b innerhalb der Menge M liegen. Analog zur Konvexität einer Funktion muss auch die Konvexitätsbedingung einer Menge für beliebige Punkte \mathbf{x}_a und \mathbf{x}_b innerhalb der Menge M gelten. Auch die Konvexität der Menge stellt hierbei ein hinreichendes aber kein notwendiges Kriterium dar. Die grafische Repräsentation der Bedingung ist in Abbildung 2.9 zu sehen. Exemplarisch sind in der Abbildung die jeweiligen Zielfunktionswerte der Isolinien dargestellt.

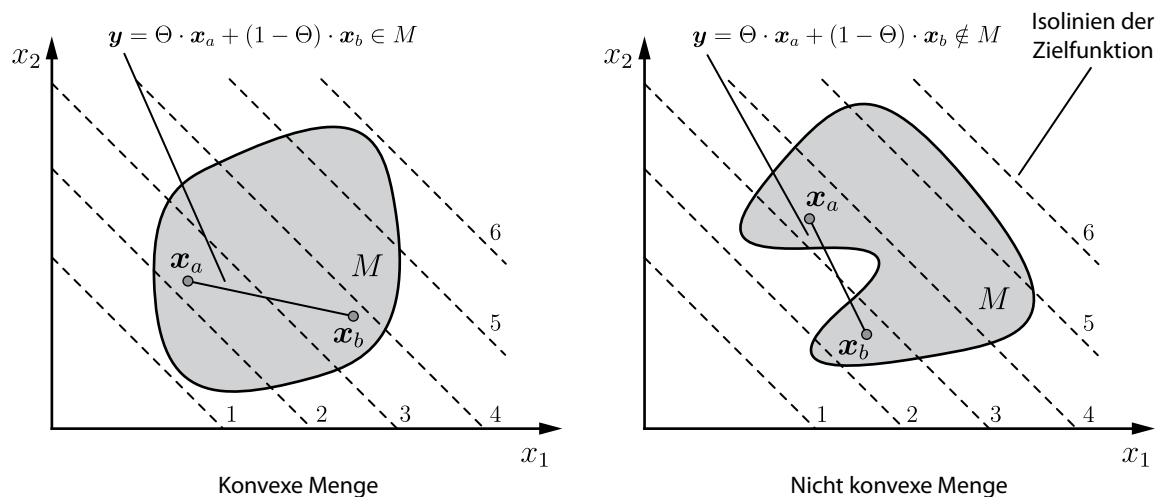


Abbildung 2.9: Konvexität einer Menge im Designvariablenraum, nach [Sch13], [Har14]

Für die Beschreibung der Konvexität einer Optimierungsaufgabe sind somit die Konvexität der Zielfunktion und die Konvexität des zulässigen Bereichs im Designvariablenraum entscheidend. Wenn die Optimierungsaufgabe konvex ist, existiert nur ein Minimum, welches das globale Minimum darstellt [Sch13]. Für eine weiterführende umfassende Darstellung konvexer Funktionen und Mengen wird auf [BSS06] verwiesen.

Bei der Bearbeitung einer Optimierungsaufgabe sind die Ziel- bzw. Restriktionsfunktionen in der Regel nur an einzelnen Stellen bekannt. Es liegen keine Informationen über deren gesamten Verlauf vor, wodurch keine Aussage über die Konvexität der Funktionen getroffen und das globale Optimum nicht sicher bestimmt werden kann [Har14]. Durch den Start der Optimierung an verschiedenen Startpunkten kann die Wahrscheinlichkeit, das globale Optimum zu finden, erhöht werden [JA93], [Sch13]. Es gibt jedoch kein Optimierungsverfahren, das sicherstellt, das globale Optimum zu finden. Auf Basis der bei der Optimierung ausgewerteten Stützstellen der Zielfunktion im Lösungsraum kann lediglich mit einer gewissen Wahrscheinlichkeit die Güte des erreichten Optimums bewertet werden. Eine möglichst vollständige Untersuchung des Lösungsraumes sowie eine deutliche

Konvergenz der Lösungen erhöhen die Wahrscheinlichkeit das globale Optimum zu finden. Da auch das Erreichen eines lokalen Minimums eine Verbesserung des Ausgangszustandes darstellt, ist dies ebenfalls als Erfolg zusehen, auch wenn eventuell noch eine bessere Lösung existiert [Har14].

2.3 Der Optimierungsprozess

Das Lösen einer Optimierungsaufgabe geschieht immer als iterativer Prozess. Selbst wenn kein Optimierungsalgorithmus verwendet wird und verschiedene Lösungen manuell erzeugt und getestet werden, erfolgt dies iterativ. Diesem iterativen Prozess liegt die Vorgehensweise des Test-Operate-Test-Exit (TOTE) zugrunde, welches einen Rückkopplungskreis als Grundelement des menschlichen Verhaltens beschreibt [MGP60]. Die ursprüngliche Version der sog. TOTE-Einheit ist in Abbildung 2.10 dargestellt.

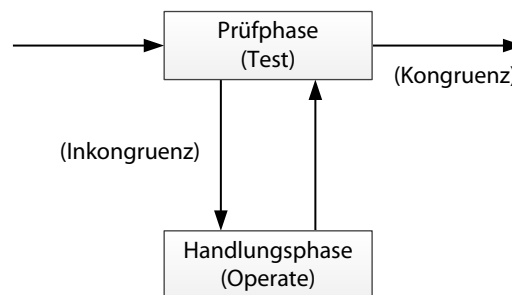


Abbildung 2.10: Ursprüngliche Version des TOTE-Schemas, nach [MGP60], [MGP91]

Das TOTE-Schema besteht aus einer Prüfphase (Test) und einer Handlungsphase (Operate), welche iterativ wie folgt ablaufen:

- **Test:** Zunächst wird ein Eingangstest durchgeführt, in dem der aktuelle Ist-Zustand mit dem Soll-Zustand verglichen wird. Wird bei diesem Vergleich Inkongruenz festgestellt, erfolgt die Handlungsphase.
- **Operate:** In der Handlungsphase wird auf Basis des Ist-Soll-Vergleichs der Ist-Zustand verändert.
- **Test:** Nach der Handlungsphase erfolgt wieder ein Test, indem das Ergebnis der Handlung überprüft wird. Es findet also wieder ein Vergleich zwischen dem Ist-Zustand und dem Soll-Zustand statt. Wird hierbei erneut Inkongruenz festgestellt, erfolgt eine weitere Handlungsphase.
- **Exit:** Verläuft der Test des Handlungsergebnisses positiv und Ist-Zustand und Soll-Zustand sind kongruent zueinander, wird der Prozess beendet.

Das Auflösen der Prüfphase in einen Entscheidungsprozess bildet die Grundlage für die Transformation des TOTE-Schemas in ein technisches Prozessschaubild, welches die Basis für Handlungsabfolgen und Algorithmen liefert und durch Ehrlenspiel [Ehr95] als grundlegende Vorgehensweise des Problemlösens in der Produktentwicklung beschrieben wird. Das Prozessschaubild des TOTE-Schemas ist in Abbildung 2.11 dargestellt.

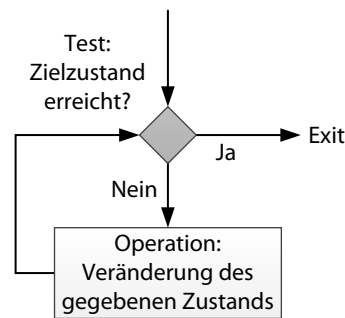


Abbildung 2.11: TOTE-Schema als Prozessschaubild, nach [Dör87], [Ehr95]

Das Lösen einer Optimierungsaufgabe mit Hilfe eines Optimierungsalgorithmus besteht im Wesentlichen aus einer endlichen Anzahl von Iterationen, in denen ein gegebener Zustand, welcher durch die Designvariablen definiert ist, verändert und anschließend geprüft wird, ob dadurch ein definierter Zielzustand erreicht wurde. Dieser iterative Prozess wird in der Regel solange wiederholt, bis der Zielzustand erreicht und somit das Optimum gefunden ist.

2.4 Designvariablen- und Lösungsraum im Optimierungsprozess

Der Ursprung einer Optimierung ist der Designvariablenraum. Er stellt den Bereich dar, in dem die Designvariablen durch den Optimierungsalgorithmus verändert werden können und wird in der Regel durch die oberen und unteren Grenzen der Designvariablen begrenzt (s. Abschnitt 2.1). Der Designvariablenraum X beschreibt somit die Menge der zulässigen Designvariablen.

Das Gegenstück des Designvariablenraumes auf der Ergebnisseite der Zielfunktionsformulierung ist der Lösungsraum F . Er bildet die Menge zulässiger Lösungen aus der Menge der zulässigen Designvariablenwerte ab:

$$F = \mathbf{f}(X) \quad (2.8)$$

Der Lösungsraum F kann somit durch die Berechnung der Zielfunktionswerte aller Punkte des Designvariablenraumes X ermittelt werden. Da die Zielfunktion in der Regel im Vorfeld einer Optimierung nicht bekannt ist, erfolgt die Ermittlung eines Zielfunktionswertes bzw. -vektors $\mathbf{f}(\mathbf{x})$ aus dem Vektor der Designvariablen \mathbf{x} durch die Evaluation der Designvariablen. Die Schnittstelle zwischen dem Designvariablenraum und dem Lösungsraum bildet somit das Evaluationsmodell.

Abbildung 2.12 gibt einen grafischen Überblick der beschriebenen Zusammenhänge. Da der Lösungsraum einer monokriteriellen Optimierung nur eindimensionalen Charakter aufweist, wird zum besseren Verständnis der zweidimensionale Lösungsraum einer bikriteriellen Optimierung dargestellt, bestehend aus den Zielfunktionen $f_1(\mathbf{x})$ und $f_2(\mathbf{x})$, welche hier minimiert werden und den Zielfunktionsvektor $\mathbf{f}(\mathbf{x})$ bilden. Multikriterielle Optimierungsstrategien werden genauer in Abschnitt 2.8.1 beschrieben.

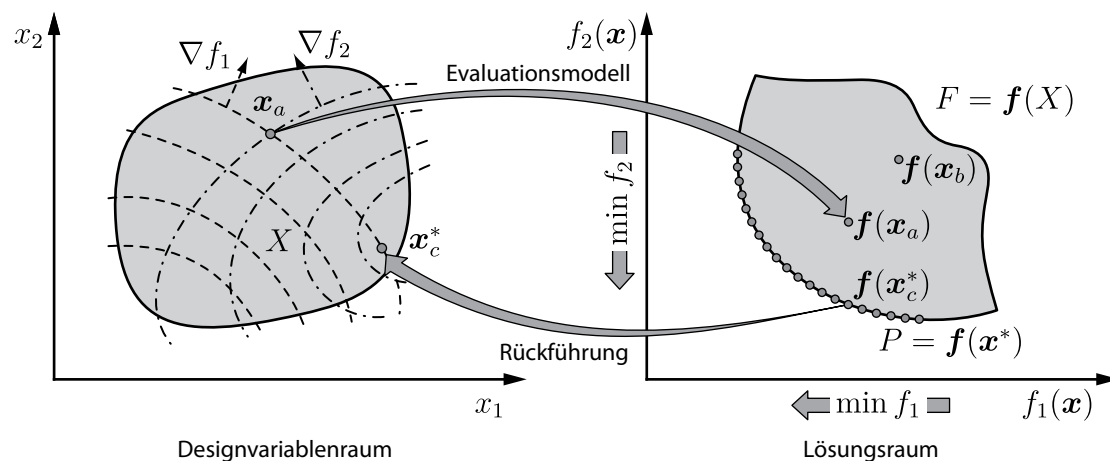


Abbildung 2.12: Abbildung des Lösungsraumes aus dem Designvariablenraum und Rückführung der Ergebnisse, nach [Bes06], [Kin11]

Wird die Validität des Evaluationsmodells vorausgesetzt, kann jedem Punkt \mathbf{x} im Designvariablenraum ein Punkt $\mathbf{f}(\mathbf{x})$ im Lösungsraum zugeordnet werden. Ist das Evaluationsmodell nicht valide, kann es nicht alle Punkte des Designvariablenraumes abbilden und Designvariablenraum und Lösungsraum sind nicht kongruent zueinander. In beiden Fällen enthält der Lösungsraum jedoch ausschließlich zulässige Varianten [Fen14].

Voraussetzung für eine hohe Wahrscheinlichkeit, das globale Optimum eines Problems zu finden, ist in jedem Fall ein möglichst großer Lösungsraum. Dies wird durch einen möglichst großen Designvariablenraum erzielt, aber auch durch ein valides Evaluationsmodell, mit welchem alle Punkte des Designvariablenraumes abgebildet werden können. Neben der Flexibilität des Evaluationsmodells ist auch dessen Güte zur Beurteilung der Validität entscheidend. Die Güte gibt an, wie gut die realen Zusammenhänge durch das Modell abgebildet werden, z. B. bei der Vorhersage der Spannungen durch eine FEM-Berechnung. Somit wird sichergestellt, dass die durch die Optimierung erzielten Ergebnisse auch auf die Realität übertragen werden können.

Insbesondere bei multikriteriellen Optimierungsaufgaben ist die Trennung von Designvariablen- und Lösungsraum sinnvoll und erforderlich, da anhand eines Zielfunktionswertes nicht eindeutig festgestellt werden kann, ob der dazugehörige Designvariablenvektor das Optimum darstellt [Bes06], [Kin11]. Das Optimum wird hierbei nicht durch einen Punkt im Lösungsraum repräsentiert, sondern durch eine Menge Pareto-optimaler Lösungen P . Zur Ermittlung dieser Menge wird ausgehend von einem Punkt \mathbf{x}_a im Parameterraum der zugehörige Vektor der Zielkriterien $\mathbf{f}(\mathbf{x}_a)$ durch Evaluation bestimmt und solange durch den Algorithmus optimiert, bis eine Verbesserung eines Zielkriteriums nur durch die Verschlechterung eines anderen Zielkriteriums möglich ist. In diesem Fall ist der optimierte Punkt $\mathbf{f}(\mathbf{x}_c^*)$ Teil der Menge Pareto-optimaler Lösungen P , welche auch als Pareto-Front bezeichnet wird.

Durch Rückführung des Punktes $\mathbf{f}(\mathbf{x}_c^*)$ aus dem Lösungsraum in den Designvariablenraum kann der dazugehörige Designvariablenvektor \mathbf{x}_c^* bestimmt werden. Da der Punkt \mathbf{x}_c^* den gleichen Wert des Zielkriteriums $f_1(\mathbf{x})$ wie der Punkt \mathbf{x}_a liefert, können durch die Rückführung die Isolinien der einzelnen Zielfunktionen in den Designvariablenraum projiziert werden. Alle Punkte auf einer Isolinie im Designvariablenraum führen zu gleichen Werten der jeweiligen Zielfunktion (s. Abschnitt 2.2). Die Isolinien der Zielfunktionen sind

in der Abbildung durch Strich- und Strichpunktlinien, die Gradienten der Zielfunktionen exemplarisch durch die Terme ∇f_1 und ∇f_2 dargestellt.

Wird die Rückführung für alle Punkte der Pareto-Front durchgeführt, kann die Pareto-Front ebenfalls in den Designvariablenraum projiziert werden. Aufgrund der Übersichtlichkeit wird in Abbildung 2.12 auf die Darstellung die Pareto-Front im Designvariablenraum verzichtet. Weitere Strategien der multikriteriellen Optimierung werden in Abschnitt 2.8.1 beschrieben.

2.5 Optimierungsmethoden

Die Auswahl der Optimierungsmethode ist ein wesentlicher Schritt in der Bearbeitung einer Optimierungsaufgabe und der Festlegung einer Optimierungsstrategie und hat somit einen großen Einfluss auf das zu erwartende Ergebnis. In den folgenden Abschnitten werden die in der Produktentwicklung angewendeten Optimierungsmethoden beschrieben. Die Unterscheidung der Optimierungsmethoden erfolgt hierbei zum einen nach der Optimierungsdisziplin, was der konventionellen Unterscheidung entspricht (s. Abschnitt 2.1). Zum anderen wird nach der Definition der Designvariablen gemäß Abbildung 2.5 unterschieden. Tabelle 2.1 gibt einen Überblick über die in der Produktentwicklung angewendeten Optimierungsmethoden.

Tabelle 2.1: Unterscheidung von Optimierungsmethoden

Optimierungsmethode	Designvariablen	Ergebnis
Parameteroptimierung	Alle Parameterarten	Parametersatz, CAD-Modell, FE-Modell, ...
Topologieoptimierung	Normierte Dichte der FE	FE-Modell
Formoptimierung	Knotenkoordinaten der FE	FE-Modell
Freie Dimensionierung	Elementdicke der FE	FE-Modell

Die Reihenfolge der Auflistung der Optimierungsmethoden erfolgt mit abnehmender Flexibilität in der Anwendung der jeweiligen Methode. So können durch die Parameteroptimierung je nach Parametrisierung und verwendetem Evaluationsmodell verschiedene Optimierungsdisziplinen abgedeckt werden. Dies zeigt sich auch anhand der Designvariablen und in dem zu erwartenden Ergebnis, welche je nach Optimierungsmethode verschiedenartig sind. Bei der Parameteroptimierung können prinzipiell alle Arten von Parametern als Designvariablen verwendet werden, z. B. Parameter eines CAD-Modells oder finiter Elemente (FE). Je nach Evaluationsmodell kann das Ergebnis einer Parameteroptimierung dann als simpler Parametersatz, CAD-Modell, FE-Modell oder in Form von anderen Modellen vorliegen.

Die weiteren Optimierungsmethoden sind sehr auf ihren Anwendungsfall spezialisiert und basieren auf dem FE-Modell des zu optimierenden Bauteils. Das FE-Modell repräsentiert

dabei die geometrische Beschreibung und die Analyse des Bauteils. Die verwendeten Modelle können zwar sehr flexibel variiert werden, da sie nicht von der Parametrisierung des Anwenders abhängen, es werden jedoch ausschließlich die jeweiligen spezifischen Optimierungsdisziplinen abgedeckt. Auch das Ergebnis der Optimierungsmethoden stellt ein zum Ursprungszustand verändertes FE-Modell dar, welches die optimierte Geometrie des Bauteils beinhaltet und anschließend eine manuelle Umsetzung in ein CAD-Modell erfordert. Zur Unterstützung der Umsetzung sowie zur direkten Weiterverarbeitung des FE-Modells existieren bereits erste Ansätze, welche ebenfalls in den folgenden Abschnitten vorgestellt werden.

Aufgrund der Komplexität des Themas sowie den vielen verschiedenen teilweise sehr spezialisierten Verfahren der Optimierungsmethoden wird hierbei nur ein Überblick gegeben. Für weiterführende Informationen wird auf einschlägige Literatur zu diesem Thema verwiesen, z. B. [BS04], [Sch13], [Har14].

2.5.1 Parameteroptimierung

Die Eigenschaften eines zu optimierenden Systems bzw. Modells werden durch Parameter und Zustandsvariablen erfasst. Hierbei repräsentieren die Parameter die definierenden Eigenschaften und die Zustandsvariablen die sich daraus ergebenden beschreibenden Eigenschaften des Systems. Die Parameter müssen dabei nicht zwangsläufig konstant sein, sondern können auch in Parameterstudien oder Optimierungsaufgaben variiert werden. Sie werden dann durch den Anwender explizit als Designvariablen definiert. Erfolgt eine einzelne Analyse des Systems bzw. Modells müssen die Parameter vom Anwender festgelegte Werte annehmen [VWBZ09].

Der Begriff der Parameteroptimierung beschreibt in dieser Arbeit Optimierungsmethoden, bei denen eine explizite Definition von Parametern als Designvariablen durch den Anwender erfolgt. Dabei können verschiedene Optimierungsverfahren und Evaluationsmodelle verwendet werden. Das Evaluationsmodell kann in analytischer Form oder in numerischer Form vorliegen. Die Parameteroptimierung stellt somit die umfassendste Methode der Optimierung dar, aufgrund dieser Vielseitigkeit aber auch die komplexeste [RMS98].

Neben der Verwendung analytischer Gleichungen als Ziel- und Restriktionsfunktionen, welche oft in den frühen konzeptionellen Phasen der Produktentwicklung oder zur Optimierung auf Basis von Metamodellen Anwendung finden, hat sich die Verwendung parametrischer CAD-Modelle zur Modellgenerierung etabliert. Das CAD-System beinhaltet somit das Entwurfsmodell. Die Analyse des Modells, z. B. durch eine FEM-Simulation, erfolgt separat, z. B. in einem CAE-System. Diese Programmtrennung ist nicht zwingend notwendig, da moderne CAx-Systeme bereits einen großen Funktionsumfang, auch zur Simulation, abdecken und somit alle Teile des Evaluationsmodells realisieren können. Sowohl bei der Verwendung verschiedener Systeme oder eines Systems mit verschiedenen Modulen findet eine *Transformation* des Modells statt. Das CAD-Modell definiert das zu optimierende Produkt und bildet die Basis für die weiteren Modelle [Die14].

Eine weitere Möglichkeit der CAD/CAE-Integration ist die Nutzung einer *gemeinsamen Datenbasis*, aus der alle benötigten Modelle generiert werden. Diese Datenbasis kann ein neutrales featurebasiertes Modell darstellen, welches die benötigten geometrischen Modelle für das CAD- und das CAE-System bereitstellt [Lee05], oder eine gemeinsame generische Parametrik, bestehend aus topologischen und geometrischen Parametern sowie

Modellattributen. Transformationsmodule bilden die Schnittstelle zu den jeweiligen parametrischen Modellen, welche gleichwertig sind [MLV⁺09]. Die Datenbasis lässt sich auch aus bereits erstellten parametrischen Modellen ableiten. Diese sehr spezifische Form der CAD/CAE-Integration findet vor allem in Konzeptentwicklung von Fahrzeugen Anwendung [DZ13].

Bei der *Isogeometrischen Analyse* erfolgt im Gegensatz zu den ersten beiden Methoden keine strikte Modelltrennung, da die mathematischen Grundlagen beider Modelle fest miteinander verknüpft sind. Es findet keine klassische Diskretisierung der Geometrie durch ein FE-Netz statt, sondern das Netz basiert direkt auf den NURBS des CAD-Modells [HCB05]. Somit ist keine Konvertierung der Modelle notwendig und die Optimierung erfolgt direkt auf einem Modell [KY16]. Die beschriebenen Methoden zur CAD/CAE-Integration sind in Abbildung 2.13 schematisch dargestellt.

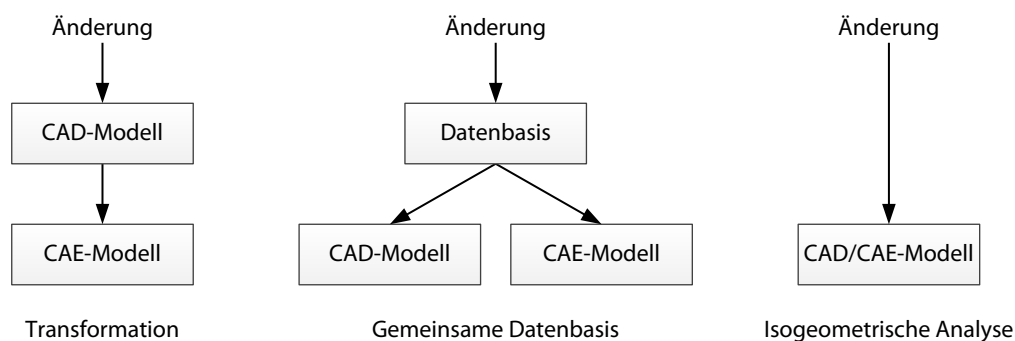


Abbildung 2.13: Schematische Darstellung der CAD/CAE-Integration, nach [Die14]

Da die Entwicklung eines Produktes nicht nur aus den Bereichen CAD und CAE besteht, sondern eine Vielzahl an verschiedenen Prozessen beinhaltet, stellen die beiden zuletzt beschriebenen Methoden zwar interessante, aber zugleich auch sehr spezialisierte Ansätze dar. Ein Großteil der Prozesse der Produktentwicklung basiert auf dem CAD-Modell. Insbesondere durch die parametrisch assoziative CAD-Modellierung können Entwicklungsmethoden strukturiert in CAD-Modellen abgebildet und Gestaltungs-, Berechnungs- und Planungsprozesse miteinander verknüpft werden. Das vorhandene Wissen lässt sich somit in den Modellen speichern und für die erneute Verwendung in neuen Produktentwicklungszyklen bereitstellen. Ein gut strukturiertes parametrisch assoziatives CAD-Modell kann weitestgehend automatisch an neue oder veränderte Referenzen anpassen werden. Nur grundlegende Topologieänderungen erfordern eine Anpassung der Parametrik oder der Assoziativität [Tec10].

Bei der parametrisch assoziativen CAD-Modellierung werden Beziehungen zwischen den zahlreichen unmittelbar festzulegenden Parametern der Systemelemente hergestellt, mit dem Ziel, die Anzahl der unabhängigen Parameter (Führungsparameter) zu reduzieren und somit den Entwicklungsprozess zu automatisieren und zu standardisieren. Weist das modellierte System eine hierarchische Struktur auf (z. B. bei einer Baugruppe), wird diese in der Regel auch auf die Parameter übertragen. Dabei können die Parameter danach unterteilt werden, ob sie Eigenschaften nur auf dieser Hierarchieebene definieren oder die für die nächst niedrigere Hierarchieebene relevanten Anforderungen spezifizieren. Auswirkungen der Parameter einer Hierarchieebene auf höhere Ebenen sollten generell vermieden werden. Hierarchieübergreifende Parameter sind auf der jeweils höchsten Ebene zu definieren und über Beziehungen mit den unteren Ebenen zu verknüpfen [VWBZ09].

Weiterhin werden die Parameter bei der parametrischen Modellierung nach der Art ihrer Erzeugung unterschieden. Parameter, welche durch den Anwender erstellt werden, werden als explizite Parameter bezeichnet, durch das System bei der Modellierung erzeugte Parameter als implizite. Moderne CAx-Systeme erzeugen diese Parameter bei allen Operationen und Eingaben. Beide Parameterarten können als Designvariablen einer Optimierung verwendet werden. Zudem können implizite Parameter auch umbenannt und weiterverarbeitet werden. Sie werden somit zu expliziten Parametern. Dieser Schritt bietet sich bei der Vorbereitung einer Parameteroptimierung an, um die Designvariablen festzulegen.

Wie eingangs erwähnt, ist die Parameteroptimierung die allgemeingültigste und vielseitigste Optimierungsmethode, mit der alle Optimierungsdisziplinen abgedeckt werden können (s. Abbildung 2.5). Dabei ist jedoch die Art der Parametrisierung und die Verknüpfung mit dem verwendeten Modell entscheidend. Die Parameter können grob in Geometrieparameter, topologische Parameter, physikalische Parameter und Prozessparameter eingeteilt werden [VWBZ09]. Wie in Tabelle 2.2 dargestellt, lassen sich diese Parameterarten den jeweiligen Optimierungsdisziplinen zuordnen. Zusätzlich werden Führungsparameter unterschieden, welche streng genommen Geometrieparameter repräsentieren, bei der Parameteroptimierung jedoch eine gesonderte Rolle einnehmen, da dadurch z. B. Steuerskizzen und Skelettmodelle sowie grundsätzliche Modelländerungen gesteuert werden.

Tabelle 2.2: Arten der Parametrisierung zur Parameteroptimierung

Opt.-Disziplin	Parameterart	Modellverknüpfung
Topologieopt.	Topologische Parameter	Indirekt (Logisch)
	Führungsparameter	Indirekt (Logisch)
Formopt.	Geometrieparameter	Direkt
	Führungsparameter	Indirekt (Geometrisch, Arithmetisch)
Dimensionierung	Geometrieparameter	Direkt
Konzeptopt.	Führungsparameter	Indirekt (Logisch)
Materialopt.	Physikalische Parameter	Direkt, Indirekt (Arithmetisch, Logisch)
	Prozessparameter	Indirekt (Arithmetisch, Logisch)
	Führungsparameter	Indirekt (Arithmetisch, Logisch)

Bei der Topologieoptimierung wird die Anordnung der das Modell beschreibenden Objekte variiert, z. B. Materialverteilung von Rippen, Stegen, Bohrungen und Hohlräumen oder die Anordnung von Komponenten einer Baugruppe. Die Anordnung dieser Objekte wird durch topologische Parameter festgelegt, welche durch logische Beziehungen indirekt mit dem Modell verknüpft sind. Dazu werden in der Regel Boole'sche Variablen (z. B. *True/False* oder *1/0*) und bedingte Anweisungen (z. B. *IF*-Anweisungen) verwendet. So kann z. B. die Existenz einer Bohrung über einen Parameter gesteuert werden. Nimmt der Parameter den Wert *True* an, existiert die Bohrung, bei dem Wert *False* wird sie unterdrückt. Alternativ ist auch eine Unterdrückung der Bohrung bei einer Bohrungstiefe oder einem Bohrungsdurchmesser von 0 denkbar. Diese Methode wird jedoch aus Gründen der Modellkonsistenz in CAD-Systemen nicht unterstützt. In diesem Fall würde durch die Bohrung zwar keine Geometrie verändert werden, das Feature-Objekt wäre jedoch trotzdem vorhanden, was z. B. bei der automatisierten NC-Programmerstellung zu

Problemen führen kann. Stellen die topologischen Parameter eines Modells nicht direkt die Designvariablen dar, können auch Führungsparameter als Designvariablen verwendet werden, welche logisch mit den topologischen Parametern verknüpft sind.

Zur Optimierung der Form können die Designvariablen direkt als Geometrieparameter oder indirekt als Führungsparameter mit dem Modell verknüpft werden. Beispielhaft für die direkte Modellverknüpfung ist die Verwendung eines Winkels zur Manipulation einer Skizzenkontur oder einer Fläche zu nennen. Die indirekte Verknüpfung von Führungsparametern mit dem Modell erfolgt meist über Steuerskizzen (geometrisch) oder analytische Formeln (arithmetisch).

Aufgrund ihrer Flexibilität und Robustheit hat sich die Verwendung von NURBS zur parametrischen Formoptimierung etabliert. Da die Definition von NURBS im Gegensatz zu Splines eine unterschiedliche Gewichtung der Kontrollpunkte zulässt, können Regelgeometrien wie Kreisbögen und Linien sowie Freiformgeometrien gleichermaßen approximiert werden [VWBZ09], [Sch13].

Werden benachbarte Konturen eines Bauteils optimiert, ist zudem die Verwendung einer Steuerskizze sinnvoll, von der ausgehend die Kontrollpunkte der NURBS variiert werden [Sch13]. Da nur positive Werte als Abstände der Kontrollpunkte zur Steuerskizze verwendet werden, können Überschneidungen und somit ein Kollabieren des CAD-Modells vermieden werden. Zudem kann die Steuerskizze ebenfalls parametrisiert und bei der Optimierung berücksichtigt werden. Abbildung 2.14 zeigt eine parametrisierte Steuerskizze und die darauf basierende Konturskizze eines Dreipunkthebels.

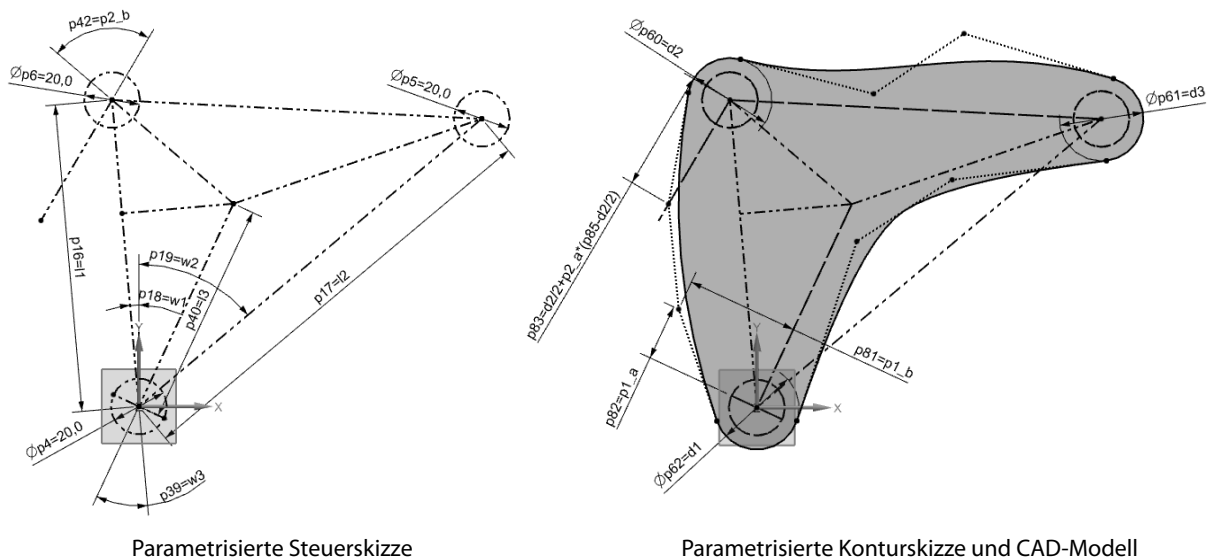


Abbildung 2.14: Parametrisierte Steuer- und Konturskizze eines CAD-Modells unter Verwendung von NURBS

Aufgrund der Übersichtlichkeit ist nur die Parametrisierung des linken NURBS der Konturskizze abgebildet. Es werden zwei unterschiedliche Arten der Parametrisierung dargestellt. Der untere Kontrollpunkt wird durch die Abstände von der Steuerskizze definiert (p_{1_a} und p_{1_b}). Diese Art der Parametrisierung kann unter Umständen zur Überlagerung der Kontrollpunkte und somit Überschneidungen des NURBS führen und erfordert eine genaue Festlegung und Überprüfung der Parametergrenzen.

Der obere Kontrollpunkt hingegen ist auf einer Linie positioniert, welche ausgehend von der Steuerskizze um einen Winkel ($p2_b$) rotiert wird. Die Position des Kontrollpunktes auf dieser Linie wird relativ zur Länge der Linie abzüglich des nichtzulässigen Bereichs aufgrund der Verrundungen definiert. Der Führungsparameter ($p2_a$) kann hierbei nur Werte zwischen 0 und 1 annehmen. Bei einem Wert von 0 befindet sich der Kontrollpunkt am Anfang der Linie, bei einem Wert von 1 am Ende. Diese Art der Parametrisierung ist wesentlich stabiler und lässt auch eine Variation der Steuerskizze zu. Die Flexibilität des mit dieser Parametrisierung erstellten Modells ist in Abbildung A.1 im Anhang A.1 dargestellt.

Neben der Variation der reinen Kontrollpunktkoordinaten der NURBS können zusätzlich auch die Richtung der Positionierungslinien, die Wichtungsfaktoren oder der Grad der NURBS als Designvariablen dienen. Die Variabilität eines Modells lässt sich durch das Einfügen weiterer Kontrollpunkte oder zusätzlicher NURBS erhöhen [Sch13].

Die Dimensionierung bildet die einfachste Form der Parameteroptimierung. Hierbei werden als Designvariablen lediglich Geometrieparameter verwendet, welche direkt mit dem Modell verknüpft sind. Dies können z. B. Wandstärken, Stegbreiten oder Bohrungsdurchmesser sein.

Sollen bei einer Optimierung verschiedene Konzepte variiert werden, erfolgt dies analog zur Topologieoptimierung durch Führungsparameter, welche über logische Beziehungen indirekt mit dem Modell verknüpft sind. Dabei können auch mehrere Teilmodelle zum Einsatz kommen, welche über die Führungsparameter in das Gesamtmodell integriert werden. Somit lassen sich verschiedene Prinziplösungen einer Entwicklungsaufgabe variieren, z. B. die Querschnitte eines Tragwerks. Die Werte des Führungsparameters repräsentieren dabei die unterschiedlichen Querschnitte, die mittels einer IF-Anweisungen im Modell aktiviert oder deaktiviert bzw. hinzugefügt oder entfernt werden.

Bei der Optimierung des Materials können zum einen physikalische Parameter direkt oder auch indirekt mit dem Modell verknüpft werden. Beispielhaft für die direkte Verknüpfung ist die Variation des Faserwinkels oder der Lagendicke faserverstärkter Kunststoffe. Die Variation verwendeter Materialien oder des Lagenaufbaus erfolgt durch indirekte Modellverknüpfung. Zudem können über indirekte Verknüpfung auch Prozessparameter aus der Fertigungstechnologie oder sonstige Führungsparameter integriert werden.

Der größte Vorteil der Parameteroptimierung liegt in der flexiblen und vielseitigen Anwendung. Die Parameteroptimierung ist nicht an ein bestimmtes Modell gebunden und kann somit in verschiedenen Phasen der Produktentwicklung eingesetzt werden. Zur Evaluation können verschiedene Modelle verwendet werden, z. B. analytische, numerische oder physikalische Modelle. Weiterhin können durch die Modelltrennung auch sehr schnell die Zielfunktion und der verwendete Optimierungsalgorithmus im Optimierungsmodell verändert und angepasst werden. Insbesondere die Verwendung parametrischer CAD-Modelle bringt weitere Vorteile mit sich, welche folgend erläutert werden:

- **Evaluationsmodelle:** Bei der Verwendung eines parametrischen CAD-Modells können verschiedene Modelle für die Analyse eingesetzt werden. Geometrische Größen wie Oberfläche, Volumen, Masse oder Trägheiten können direkt aus dem CAD-Modell ermittelt werden. Weitere Simulationen (z. B. CAE oder CAM) können unabhängig voneinander in beliebiger Komplexität verwendet und in eine multidisziplinäre Optimierung (s. Abschnitt 2.8.2) integriert werden.

- **Vorhandene Parametrik:** Vorhandene bereits parametrisierte CAD-Modelle können für eine Optimierung verwendet werden.
- **Überschneidungen und Durchdringungen:** Aufgrund ihres Modellierkerns können CAD-Systeme von Grund aus Überschneidungen und Durchdringungen verarbeiten. Zusätzliche Volumengeometrie wird in diesem Fall immer dem aktuellen Volumenkörper hinzugefügt.
- **Verarbeitung von Baugruppen:** Bei der Verwendung eines CAD-Systems ist das Modell nicht auf einzelne Bauteile oder eine Ansammlung einzelner Komponenten begrenzt. Es können ganze Baugruppen und deren Produktstruktur abgebildet werden.
- **Neuvernetzung:** Wird ein CAD-Modell in Verbindung mit einer FE-Analyse verwendet, muss in der Regel jede Variante neu vernetzt werden. Das Netz wird nicht zusätzlich gezerrt oder gestaucht und es treten keine Netzverzerrungen auf, welche das Ergebnis beeinflussen können.
- **CAD-Modell als Ergebnis:** Das Ergebnis der Optimierung liegt direkt als CAD-Modell vor. Das CAD-Modell muss nach der Optimierung also nicht manuell aus dem Optimierungsergebnis erstellt werden.

Obwohl die Parameteroptimierung auf Basis parametrischer CAD-Modelle wesentliche Vorteile mit sich bringt, sind auch folgende Nachteile zu beachten:

- **Modellerstellung:** Die Erstellung eines flexiblen und stabilen parametrischen CAD-Modells kann sehr zeitaufwendig und komplex sein, da bei der Erstellung die zu realisierenden Varianten vorgedacht werden müssen und die Stabilität des Modells getestet werden sollte.
- **Modellflexibilität:** Die Flexibilität des Modells hängt von der Art der Parametrisierung ab. Modelle zur Dimensionierung sind leicht zu erstellen, aber nicht sehr flexibel. Modelle zur Topologieoptimierung sind sehr flexibel und sehr komplex in der Parametrisierung. Zudem sollte je nach Optimierungsaufgabe geprüft werden, ob durch eine Optimierungsmethode mit impliziter Parameterdefinition eine größere Flexibilität erzielt werden kann.
- **Prozessautomatisierung und Integration der CAx-Systeme:** Der gesamte Evaluationsprozess muss automatisiert werden, was je nach Komplexität nur durch Programmierung und eigene Skripte realisiert werden kann.
- **Stabile Modellgenerierung:** Die Aktualisierung des CAD-Modells oder die Vernetzung kann aufgrund von nicht vorhergesehenen Kombination der Eingabeparameter fehlschlagen. Daher ist in jedem Fall eine stabile Fehlerbehandlung anzustreben, um ein Fehlschlagen der gesamten Optimierung zu verhindern.

Neben der klassischen parametrischen assoziativen Modellierung birgt die Verwendung vorparametrisierter Objekte großes Potential zur Parameteroptimierung in der Produktentwicklung, insbesondere bei einer großen Anzahl ähnlicher Bauteile. Ausgehend von der grundlegenden Struktur eines Bauteiles werden die Features aus einer Bibliothek dem Modell hinzugefügt. Dabei können auch topologische Änderungen und verschiedene Fertigungsverfahren abgebildet und bei der Optimierung berücksichtigt werden [SL14]. Aufgrund des hohen Modellierungsaufwandes bei der Erstellung der vorparametrisierten Objekte ist diese Methode nur bei einer großen Anzahl ähnlicher Bauteile wirklich effizient.

Weiterhin stellt die direkte Modellierung eine Alternative zur klassischen parametrischen Modellierung dar, vor allem wenn ein vorhandenes Modell im Nachhinein parametrisiert werden soll und die Modellierungshistorie nicht mehr zur Verfügung steht. Die Modellierung im CAD-System erfolgt in diesem Fall direkt auf den Kanten und Flächen des Modells, welche in Abhängigkeit von Parametern verschoben, gedreht oder gelöscht werden können. Beispielhaft sind hierzu die Funktionalitäten der *Synchronen Konstruktion* des CAx-Systems NX zu nennen [Sie15].

Weiteres Potential bei der Verwendung der direkten Modellierung wird in dem Modul *Form realisieren* des CAx-Systems NX gesehen. Hierbei werden ähnlich zu NURBS die Kontrollpunkte eines das CAD-Modell umgebenden Käfigs manipuliert und dadurch die Form des Modells verändert. Somit lässt sich sehr schnell ein flexibles Modell zur Formoptimierung erzeugen. Die Kontrollpunkte des Käfigs lassen in der aktuellen Version noch keine Parametrisierung zu. Dieser Ansatz stellt jedoch großes Potential dar.

Liegt kein CAD-Modell für eine Parameteroptimierung vor, kann diese auch direkt auf dem FE-Modell erfolgen. Dazu wird das FE-Netz in Bereiche eingeteilt, welche dann basierend auf Parametern durch Morphing manipuliert werden [Har14]. Es findet keine Neuvernetzung statt, sondern das Netz wird gestaucht oder gezerzt. Somit bleiben die Randbedingungen des FE-Modells erhalten. Vor allem große Geometrieänderungen können jedoch zu stark verzerrten Netzen führen. Aufgrund der fehlenden Geometriebeschreibung kann in diesem Fall auch eine adaptive Verfeinerung des FE-Netzes nur begrenzt verwendet werden. Das Morphing ist daher nur für einfache Manipulationen der Geometrie geeignet [Sch13]

2.5.2 Topologieoptimierung

Die Topologie eines Bauteils beschreibt die Anordnung der das Bauteil definierenden Objekte, z. B. die Lage und Anordnung von Strukturelementen wie Rippen, Stegen, Bohrungen und Hohlräumen. Das Ziel der Topologieoptimierung ist es, die optimale Bauteilstruktur innerhalb eines vorgegebenen Bereichs zu bestimmen. Ausgehend von wenigen Vorgaben wie Belastungen und Randbedingungen wird bei den meisten Topologieoptimierungsverfahren der zulässige Bauraum mit finiten Elementen vernetzt und durch Modifikation des FE-Netzes und dessen Eigenschaften die Geometrie verändert sowie das Einbringen von Löchern und Hohlräumen simuliert.

Dabei wird zwischen der mathematischen Topologieoptimierung und der empirischen Topologieoptimierung unterschieden. Bei der *mathematischen* Topologieoptimierung wird ein mathematisches Optimierungsproblem gelöst, indem die Zielfunktion unter Berücksichtigung von Restriktionen minimiert wird. Hierbei kann z. B. der E-Modul der finiten Elemente über eine genormte Dichte variiert werden.

Die *empirische* Topologieoptimierung basiert auf empirischen Iterationsvorschriften, nicht auf der mathematischen Formulierung des Optimierungsproblems. Dabei wird z. B. beim SKO-Verfahren (Soft Kill Option) die adaptive biologische Wachstumsregel simuliert, um eine Lösung mit homogener Oberflächenspannung zu erhalten [Har14].

Aufgrund der numerischen Stabilität der FE-Berechnung wird bei der häufig eingesetzten mathematischen Topologieoptimierung der E-Modul der Elemente nicht auf 0, sondern lediglich auf sehr kleine Werte reduziert. Somit bleiben die finiten Elemente im Modell

vorhanden, simulieren jedoch das mechanische Verhalten von Löchern und Hohlräumen [BS04], [Sch13], [Har14]. Da der E-Modul jedes Elementes eine Designvariable der Optimierung repräsentiert, ist die Topologieoptimierung gleichbedeutend mit einer sehr großen Anzahl an Designvariablen, welche nicht einzeln durch den Anwender definiert werden können, sondern auf Basis der Bauraumdefinition implizit durch das System festgelegt werden. Diese Methode wird daher auch als Pixel-Methode bezeichnet [Sch13].

Als Zielkriterien oder Restriktionen werden einfache Zustandsvariablen, wie das Volumen, das Gewicht oder die mittlere Nachgiebigkeit des Modells verwendet. Aufgrund der einfachen Formulierung des Optimierungsproblems kann die Topologieoptimierung sehr schnell angewendet werden. Zudem können auch komplexere Zielkriterien oder Restriktionen wie die maximalen Spannungen verwendet werden [HTK13]. Untersuchungen zeigen jedoch, dass die Modellvariante mit maximaler Steifigkeit (also mit minimaler mittlerer Nachgiebigkeit) identisch zur Variante mit homogener Dehnungsenergie-dichte auf der Oberfläche ist, was bei isotropen inkompressiblen Materialien zu einer homogenen Oberflächenspannung führt. Daraus resultiert die zulässige Vereinfachung der Verwendung der mittleren Nachgiebigkeit anstelle der maximalen Spannungen als Optimierungsziel. Diese Vereinfachung gilt jedoch nur bei einem Lastfall oder wenn ein Lastfall die anderen dominiert [Ped00] [Har14].

Da die Topologieoptimierung rein auf dem FE-Modell eines Bauteils basiert, wird nicht zwischen Geometrie- und Analysemodell unterschieden. Das FE-Modell beinhaltet die Geometrie- bzw. die Topologiebeschreibung und die Analyse [FFBH12]. Dies hat zur Folge, dass auch das Ergebnis der Optimierung lediglich als FE-Modell vorliegt. Das Ergebnis einer Topologieoptimierung ausgehend von einem zweidimensionalen FE-Modell des zur Verfügung stehenden Bauraumes ist in Abbildung 2.15 dargestellt. Der Bauraum ist am oberen Rand fest eingespannt und wird senkrecht durch eine Kraft belastet. Das Ziel der Optimierung ist die Minimierung der mittleren Nachgiebigkeit, also eine Maximierung der Steifigkeit. Gleichzeitig darf das optimierte Modell nur 30 % des zur Verfügung stehenden Bauraumes einnehmen.

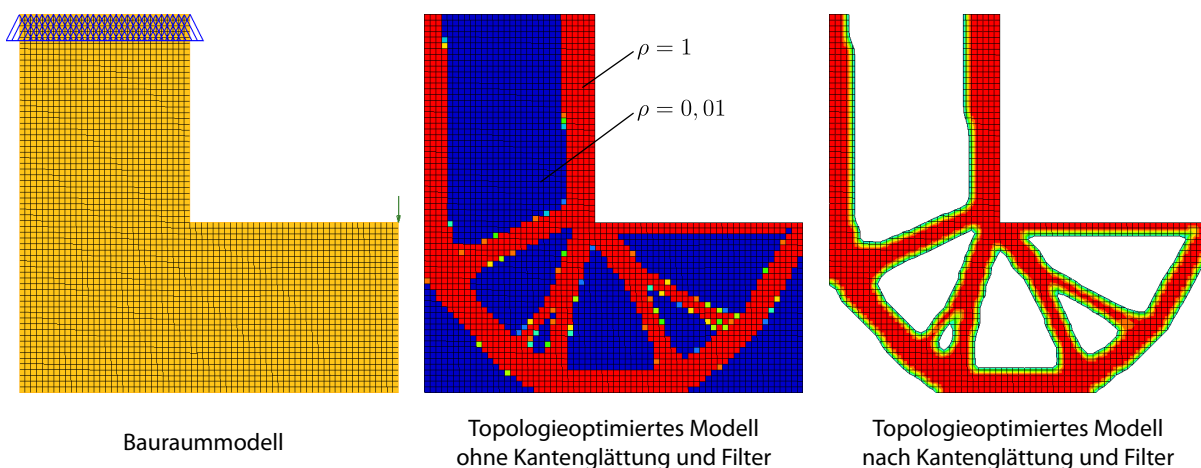


Abbildung 2.15: Topologieoptimierung eines biegebeanspruchten Hakens

Die dargestellte Topologieoptimierung erfolgt nach dem SIMP-Ansatz (Solid Isotropic Material with Penalization). Dabei wird nicht direkt der E-Modul der finiten Elemente als Designvariablen der Optimierung verwendet, sondern ein normierter Dichtewert ρ ,

über den nach der folgenden Gleichung die Steifigkeitsmatrix K_i jedes finiten Elementes reduziert bzw. bestraft wird [Alt14].

$$\underline{K}_i(\rho_i) = \rho_i^p \cdot K_i \quad \text{mit } p > 1; \quad 0 < \rho_i < 1 \quad (2.9)$$

Das Ergebnis ist die bestrafte Steifigkeitsmatrix \underline{K}_i , wodurch eine reduzierte Steifigkeit des Elementes erzielt und somit ein Loch in der Struktur simuliert wird. Da die normierte Dichte ρ kontinuierlich zwischen 0 und 1 variiert wird, können auch Elemente erzeugt werden, welche sich nicht eindeutig einem Loch oder einem festen Bereich zuordnen lassen. Daher wird der Straffaktor p verwendet. Je größer dieser Faktor ist, desto eher werden die Elemente einem Dichtewert von 0 oder 1 zugeordnet. Hierin liegt der Vorteil des SIMP-Ansatzes. Ein weiterer Vorteil ist die numerische Stabilität, da an den Übergängen zwischen Löchern und festen Bereichen keine künstlichen Kerben zwischen den finiten Elementen entstehen, sondern Elemente reduzierter Steifigkeit verwendet werden. In kommerziellen Programmen werden in der Regel Straffaktoren zwischen 2 und 4 verwendet. Für eine eindeutige Elementzuordnung wird ein Wert größer als 3 empfohlen [Har14].

Wie in Abbildung 2.15 zu sehen ist, kann die Interpretation des Ergebnisses durch die Verwendung von Kantenglättung und Filter vereinfacht werden. Dazu wird in der Nachbearbeitung des Ergebnisses für jedes Element ein Dichteverlauf ausgehend von dessen benachbarten Elementen berechnet, was zu einer Glättung der Kanten des Modells führt. Die Bereiche unterhalb eines bestimmten Dichtewertes können dann ausgeblendet werden. In der Abbildung sind die Bereiche unterhalb eines Wertes von 0,3 ausgeblendet.

Soll nur ein bestimmter Bereich eines Bauteils optimiert werden, können in dem FE-Modell des Bauraumes, sog. Non-Design-Bereiche, definiert werden. Auf diese Weise bleiben die Anschlussflächen des Bauteils erhalten. Abbildung 2.16 zeigt das dreidimensionale Bauraummodell und das optimierte Modell unter der Verwendung von Non-Design-Bereichen, einer Kantenglättung und eines Dichtefilters von 0,3. Das Optimierungsziel ist ebenfalls die Maximierung der Steifigkeit unter der Restriktion, dass das optimierte Modell nur 30 % des zur Verfügung stehenden Bauraumes einnehmen soll.

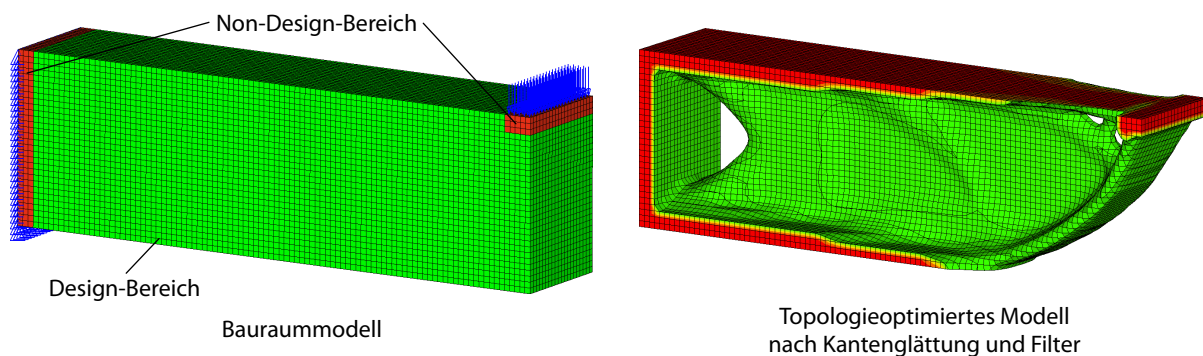


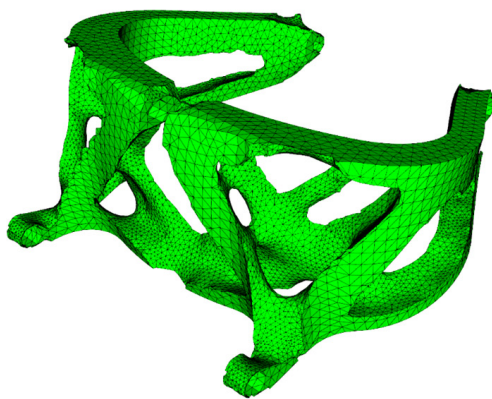
Abbildung 2.16: Topologieoptimierung eines biegebeanspruchten Balkens

Zusätzlich können bei der Topologieoptimierung Fertigungsrandbedingungen berücksichtigt werden, wie minimale und maximale Stegbreiten und Wandstärken, Symmetrien, das Vermeiden von Löchern sowie die Definition einer Auszugsrichtung, wodurch Hinterschnitte in der Struktur vermieden werden. Das in Abbildung 2.16 dargestellte Optimierungsergebnis beinhaltet die Symmetrie entlang der Längsebene und eine Auszugsrichtung quer zu dieser Ebene.

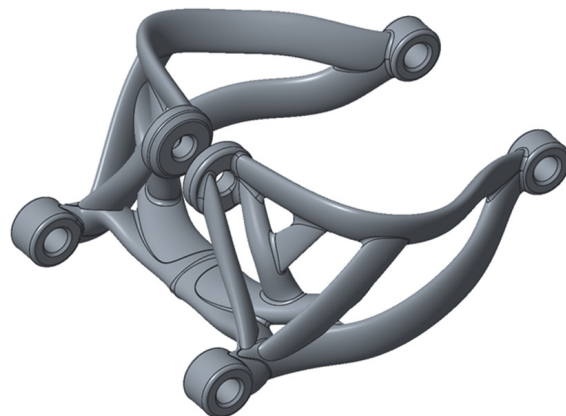
Anhand der Ergebnisse ist zu erkennen, dass die Topologieoptimierung vor allem Konstruktionsvorschläge eines Bauteils liefert, welche dann vom Anwender interpretiert werden. Da die Bauteilstruktur das mechanische Verhalten zudem entscheidend beeinflusst, sollte die Topologieoptimierung in einer sehr frühen Phase der Produktentwicklung durchgeführt werden [Sch13]. Zudem ist die Topologieoptimierung insbesondere für die Entwicklung von Gussteilen sinnvoll, da sich die optimierte Geometrie ohne wesentliche Veränderungen realisieren lässt [HG05], [HG06]. Auch durch additive Fertigungstechnologien können die Optimierungsergebnisse ohne große Anpassungen umgesetzt werden. Nach der Glättung der Kanten kann das Ergebnismodell exportiert und direkt additiv gefertigt werden. Somit ist nicht zwangsläufig ein CAD-Modell notwendig [ZP16].

Für den Einsatz konventioneller Fertigungstechnologien ist in der Regel ein CAD-Modell notwendig, da das FE-Modell lediglich eine diskrete Repräsentation der Geometrie darstellt. Die Erzeugung des CAD-Modells erfolgt manuell auf Basis des Optimierungsergebnisses. Dieser Prozess kann jedoch durch verschiedene Ansätze automatisiert unterstützt werden.

Ein Ansatz ist es, die wesentlichen Topologiepfade aus dem FE-Modell zu extrahieren. Hierzu wird das Ergebnismodell soweit geschrumpft, bis die Pfade nur noch als Drahtmodell vorhanden sind. Das Drahtmodell wird dann in das CAD-System importiert und durch Splines ersetzt. Anschließend erfolgt die Generierung des Volumenmodells durch Extrusionen entlang dieser Splines. Die Generierung des CAD-Modells erfolgt teilautomatisch. Die Topologiepfade werden automatisch nachgebildet. Die Anschlussgeometrie wie z. B. Bohrungen müssen weiterhin manuell erstellt werden [SW15]. Das Ausgangsmodell und das Ergebnis dieses Verfahrens sind in Abbildung 2.17 dargestellt.



Ergebnis der Topologieoptimierung



Teilautomatische Generierung des CAD-Modells

Abbildung 2.17: Teilautomatische Generierung des CAD-Modells aus dem Ergebnis einer Topologieoptimierung, nach [SW15]

Eine weitere Möglichkeit ist die Integration der Funktionalitäten der Topologieoptimierung in das CAD-System. Hierbei werden bisher lediglich zweidimensionale Strukturen unterstützt. Die Integration von dreidimensionaler Geometrie wird bei der Weiterentwicklung dieses Ansatzes weiterverfolgt [SST15].

Neben dem beschriebenen SIMP-Verfahren existieren weitere Verfahren zur Topologieoptimierung, wie das SKO-Verfahren, die Level-Set-Methode oder diverse Weiterentwicklungen und Speziallösungen. Diese Verfahren basieren dabei immer auf FE-Netzen und

unterscheiden sich auch sonst nur wenig voneinander. Obwohl die Topologieoptimierung eine sehr gute und einfach anzuwendende Optimierungsmethode darstellt, bleibt aufgrund der vielen verschiedenen Verfahren und Einstellungsmöglichkeiten die Frage der Lösungsbewertung und des besten Optimierungsverfahrens offen [SM13].

2.5.3 Formoptimierung

Alternativ zur Formoptimierung auf Basis parametrischer CAD-Modelle oder durch Morphing (s. Abschnitt 2.5.1) kann die Formoptimierung auch direkt auf dem FE-Netz erfolgen. Hierbei stellen die Koordinaten der FE-Knoten an der Bauteiloberfläche die Designvariablen für die Optimierung dar. Ähnlich wie bei der Topologieoptimierung führt die Verwendung der FE-Knoten zu einer sehr großen Anzahl an Designvariablen, welche nicht durch den Anwender, sondern durch das System, also implizit, auf Basis der Knotenauswahl definiert werden. Somit ergibt sich eine große Flexibilität in der Geometrievariation. Daher wird diese Methode auch als *Freiformoptimierung* bezeichnet. Aufgrund der großen Anzahl an Designvariablen werden bei dieser Methode meist gradientenbasierte Optimierungsverfahren eingesetzt [Har14].

Da die freie Variation eines Knotens unabhängig von dessen benachbarten Knoten zu enormen Verzerrungen des FE-Netz führen würde, werden die Knotenverschiebungen durch verschiedene Randbedingungen gesteuert, z. B. durch Basisvektoren oder die Vorgabe einer definierten Verschiebungsrichtung oder -größe sowie andere Regularisierungstechniken [FFBH12]. Die Generierung der Basisvektoren erfolgt in der Regel durch das System, indem die verschiedenen Lastfälle des FE-Modells berechnet werden und die jeweils resultierende Verformung als Basisvektor gespeichert wird [Sch13]. Weitere Randbedingungen können durch das System und den Anwender definiert werden. Da bereits die Verschiebung der Knoten an den Modellkanten (2D) bzw. an der Modelloberfläche (3D) zu starken Netzverzerrungen führen kann, werden auch die FE-Knoten der darunter liegenden Elementreihen bzw. -schichten verschoben, um somit die Verzerrungen gleichmäßig auf das Netz zu verteilen.

Mit Hilfe der Formoptimierung sind keine wesentlichen Änderungen der Bauteilstruktur zu erzielen, sondern in erster Linie lokale Variationen der Bauteiloberfläche. Daher wird diese Methode oft für spannungsdominierte Probleme verwendet. Dabei kann zum einen die maximale Spannung gemäß der zulässigen Knotenverschiebungen minimiert werden oder eine Minimierung der Masse bzw. des Volumens bei gleichzeitiger Spannungsrestriktion erfolgen. Abbildung 2.18 zeigt das Ausgangsmodell und das spannungsminimierte Ergebnis der Formoptimierung einer zugbeanspruchten Platte.

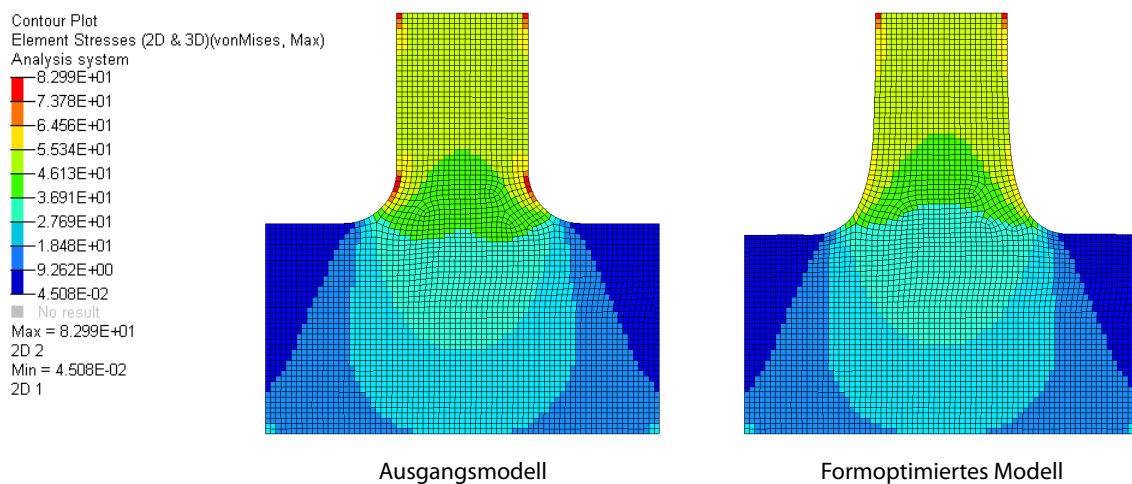


Abbildung 2.18: Formoptimierung einer zugbeanspruchten Platte

Anhand des optimierten Modells ist zu erkennen, dass sich die Kontur der Kerbe von einem einfachen Kreisbogen zu einer spannungsreduzierten Kontur nach bionischem Vorbild ändert [AEH⁺10]. Die Vergleichsspannung nach v. Mises kann somit von 75,8 MPa auf 57,3 MPa reduziert werden, was einer Reduzierung um 24,4 % entspricht.

Da bei dieser Formoptimierung die FE-Knoten an der gesamten Kontur auf beiden Seiten als Designvariablen verwendet werden, kommt es zu einer Absenkung der waagerechten Kanten. Somit werden durch das System zusätzliche Unstetigkeiten in der Kontur aufgrund der Knotenverschiebung verhindert. Sollen die waagerechten Bereiche nicht verändert werden, müssen zusätzliche geometrische Randbedingungen an den Übergängen zwischen festen und verschiebbaren Knoten definiert werden.

Eine Sonderform der Formoptimierung ist die Sickenoptimierung, welche auch als *Topographieoptimierung* bezeichnet wird. Hierbei werden die Knoten zweidimensionaler FE-Elemente nicht in der Elementebene, sondern senkrecht dazu verschoben, wodurch Sicken in das Modell eingebracht werden. Die Variation der Form des Modells ist ebenfalls nur gering. Vom Anwender werden neben dem FE-Modell im Ausgangszustand lediglich grundlegende Einstellungen wie die minimale Sickenbreite, die maximale Sickentiefe oder der Ziehwinkel getroffen. Aus diesen Vorgaben werden durch das System Basisvektoren zur Erzeugung der Sicken generiert. Bei der Optimierung wird dann die optimale Anordnung dieser Basisvektoren ermittelt. Das Ziel der Sickenoptimierung ist in den meisten Fällen die Maximierung der Steifigkeit oder der Eigenfrequenzen einer dünnwandigen Struktur. Abbildung 2.19 zeigt die Sickenoptimierung einer Ölwanne mit dem Ziel der Maximierung der Eigenfrequenzen. Weiterhin ist das Ergebnismodell nach der automatischen Sickenerstellung und Kantenglättung dargestellt.

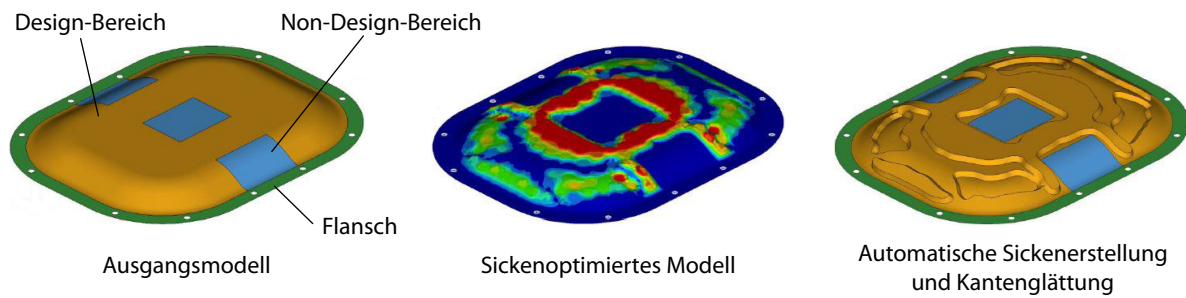


Abbildung 2.19: Sickenoptimierung einer Ölwanne, nach [Alt15]

Durch das Einbringen der Sicken erhöht sich die Steifigkeit der Ölwanne bei nahezu gleicher Masse, was zu einer Erhöhung der Eigenfrequenzen führt. So kann die erste Eigenfrequenz durch die Sickenoptimierung von 619 Hz auf 1551 Hz erhöht werden. Da nicht alle Sicken des optimierten Modells realisiert werden können, wird die erste Eigenfrequenz durch die automatische Sickenstellung und Kantenglättung auf 1422 Hz reduziert. Die erste Eigenfrequenz kann somit um 129 % gegenüber dem Ausgangszustand erhöht werden.

Aufgrund der großen Verformungen des FE-Netzes durch das Einbringen der Sicken kann es Elementverzerrungen und somit zu künstlichen Versteifungen des FE-Modells kommen. Diese Netzverzerrungen können durch den Einsatz von Filtern und Netzregulierungstechniken vermieden werden.

Wie die Topologieoptimierung basiert die Formoptimierung allein auf dem FE-Modell eines Bauteils, wodurch die verwendeten Modelle sehr flexibel variiert werden können, da sie nicht von der Parametrisierung des Anwenders abhängig sind. Das Ergebnis der Formoptimierung stellt somit immer ein zum Ursprungszustand verändertes FE-Modell dar, welches die optimierte Geometrie des Bauteils beinhaltet und anschließend manuell in einem CAD-Modell umgesetzt werden muss.

2.5.4 Freie Dimensionierung

Die freie Dimensionierung basiert ebenfalls auf dem FE-Modell eines Bauteils. Dabei werden die Designvariablen durch die Dicken der einzelnen Schalenelemente des FE-Netzes repräsentiert. Die Parameterdefinition erfolgt somit implizit durch das Optimierungssystem. Durch die stufenlose Variation der Dicke jedes Schalenelementes ähnelt das Ergebnis dem der Topologieoptimierung. Erreicht die Schalendicke einen Wert von 0, wird das Element entfernt und es entsteht ein Loch.

Wie in Abbildung 2.20 dargestellt, wird bei der freien Dimensionierung die Elementdicke im Gegensatz zur Topologieoptimierung innerhalb eines *kontinuierlichen* Wertebereichs variiert, was zu kontinuierlichen Wandstärkeübergängen führt. Zusätzlich können Randbedingungen und Restriktionen wie die minimale Elementdicke, Symmetrien und zulässige Spannungen berücksichtigt werden [Alt15].

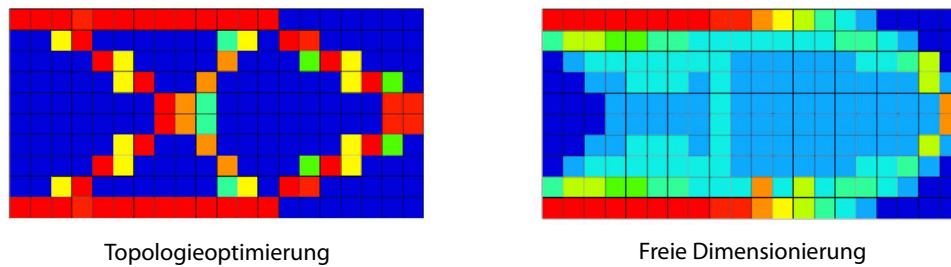


Abbildung 2.20: Ergebnisse der Topologieoptimierung und der freien Dimensionierung eines Kragträgers [Alt15]

Da das Ergebnis der freien Dimensionierung durch die variable Wandstärkeverteilung zunehmend dreidimensionalen Charakter aufweist, können ähnliche Ergebnisse auch durch eine dreidimensionale Topologieoptimierung erzielt werden. Bei dem in Abbildung 2.21 dargestellten Beispiel wird eine variable Wandstärkeverteilung durch ein Volumenmodell aus zehn Elementschichten erzielt.

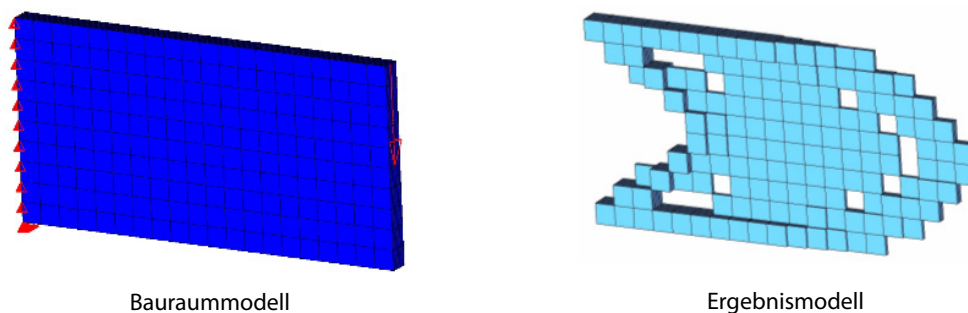


Abbildung 2.21: Topologieoptimierung eines Kragträgers unter Verwendung eines zehnschichtigen Volumenmodells [Alt15]

Der Vergleich des Ergebnisses mit dem der freien Dimensionierung in Abbildung 2.20 zeigt die Ähnlichkeit beider Methoden. Durch eine Verfeinerung der verwendeten Schichten ist eine Verfeinerung des kontinuierlichen Wandstärkeüberganges möglich. Der kontinuierliche Wandstärkeübergang der freien Dimensionierung führt zu einer größeren Gestaltungsfreiheit, was diese zu einer Alternative zur Topologieoptimierung für die Kozeptionierung macht. Die Beschränkung durch die Verwendung von 2D-Elementen stellt somit keine Beschränkung in der Formulierung des Optimierungsproblems dar [Alt15].

Auch bei dieser Optimierungsmethode repräsentiert das FE-Modell die geometrische Beschreibung und die Analyse eines Bauteils, was dazu führt, dass das Ergebnis der Optimierung als FE-Modell vorliegt und anschließend manuell in ein CAD-Modell überführt werden muss.

2.6 Optimierungsverfahren

Das Grundkonzept einer Optimierung stellt die iterative Annäherung an ein Optimum dar. Bei der Bearbeitung einer Optimierungsaufgabe können verschiedene Optimierungsverfahren verwendet werden. Optimierungsverfahren beschreiben dabei die zur Lösung der

Aufgabe verwendeten Ansätze und Optimierungsalgorithmen. Hierbei werden drei Klassen unterschieden: deterministische, stochastische und hybride Verfahren. In den folgenden Abschnitten werden die grundlegenden Eigenschaften und Funktionsweisen dieser Klassen erläutert und innerhalb der Klassen verschiedene Verfahren voneinander abgegrenzt.

Angesichts der Komplexität und der großen Anzahl teilweise sehr spezialisierter Optimierungsansätze und -algorithmen (über 1000 unterschiedliche Systeme zur Optimierung weltweit [Sch13]) wird sich in den folgenden Abschnitten auf die wichtigsten Verfahren und deren charakteristischen Eigenschaften fokussiert. Für weiterführende Informationen wird auf einschlägige Literatur zu diesem Thema verwiesen, z. B. [RS12], [BC11], [Cav13], [Sch13], [Har14], [Wei15], [SG15].

Weiterhin werden aufgrund der besonderen Eignung genetischer Algorithmen zur Lösung von diskreten und kombinatorischen Problemen, wie sie auch in dieser Arbeit bei der Optimierung der Prozessprioritäten zu finden sind (s. Abschnitt 5.4.2), die Struktur und die Anwendung genetischer Algorithmen detaillierter beschrieben.

2.6.1 Deterministische Optimierungsverfahren

Bei deterministischen Optimierungsverfahren kommen in der Regel Algorithmen zum Einsatz, welche einem strikten mathematischen Ablauf folgen und somit nur definierte und reproduzierbare Zustände liefern. Zufallsbasierte Operationen sind nicht vorgesehen. Aufgrund des strikten mathematischen Ablaufes der Verfahren wird dieses Teilgebiet der Optimierung auch als mathematische Optimierung oder mathematische Programmierung bezeichnet, wobei der Begriff der Programmierung eher als Planung bzw. Ablauf zu sehen ist [BC11]. Ein wichtiger Bestandteil vieler deterministischer Verfahren ist die Approximation der Ziel- und ggf. der Restriktionsfunktionen sowie deren Gradienten. Dabei setzen die Algorithmen meist ein konvexes Optimierungsproblem voraus (s. Abschnitt 2.2). Da die Zielfunktion und deren Konvexität im Vorfeld einer Optimierung in der Regel nicht bekannt sind, kann nicht gewährleistet werden, dass das globale Optimum gefunden wird [Cav13], [Sch13].

Deterministische Optimierungsverfahren werden nach der Verarbeitung von Restriktionen unterschieden. Dabei ist relevant, ob die Verfahren Probleme mit Restriktionen verarbeiten können. Weiterhin wird die Dimension des Optimierungsproblems bzw. die Anzahl der Designvariablen unterschieden. Obwohl eindimensionale Optimierungsprobleme (univariat) in der Realität selten zu finden sind, sind diese Verfahren auch für mehrdimensionale Optimierungsprobleme (multivariat) relevant, da z. B. bei Suchrichtungsverfahren das Optimierungsproblem auf ein oder mehrere eindimensionale Probleme reduziert wird. Bei der eindimensionalen Optimierung, welche auch als Liniensuche bezeichnet wird, haben sich zwei Verfahren etabliert: die Methode des goldenen Schnitts (Intervallreduktion) und die Polynominterpolation, welche sowohl bei der Optimierung ohne Restriktionen als auch bei der direkten Berücksichtigung von Restriktionen Anwendung finden [Sch13], [Har14].

Eine Unterscheidung der deterministischen Optimierungsverfahren zeigt die folgende Auflistung. Auf die Einordnung eindimensionaler Verfahren wird aus Gründen der Übersichtlichkeit verzichtet.

– **Optimierung ohne Restriktionen:**

- *Simplex nach Spendley* [SHH62]
- *Simplex nach Nelder und Mead* [NM65]
- Suchrichtungsverfahren 0. Ordnung:
 - *Powell* [Pow64]
- Suchrichtungsverfahren 1. Ordnung:
 - *Gradientenverfahren, Verfahren des steilsten Abstiegs (Steepest Descent)* [Cur44]
 - *Konjugierte Gradienten (Fletcher und Reeves)* [Fle64]
- Suchrichtungsverfahren 2. Ordnung:
 - *Newton-Verfahren*
 - *Quasi-Newton-Verfahren*
 - *Levenberg-Marquardt-Verfahren* [Mor78]

– **Optimierung mit Restriktionen:**

- Indirekte Verfahren:
 - *Straffunktionen*
 - *Erweitertes Lagrange-Verfahren*
- Direkte Verfahren:
 - *Methode der zulässigen Richtungen* (Method of Feasible Directions, MFD) [Zou60]
 - *Modifizierte Methode der zulässigen Richtungen* (Modified Method of Feasible Directions, MMFD) [Van84]
 - *Generalisierte Methode der reduzierten Gradienten* (Generalized Reduced Gradient Method) [GR77]

Bei Optimierungsverfahren ohne Restriktionen werden in erster Linie Suchrichtungsverfahren gemäß ihrer Ordnung unterschieden. Diese Verfahren arbeiten in mehreren Iterationen mit einer Bestimmung der Suchrichtung, gefolgt von einer eindimensionalen Optimierung. Die Suchrichtungsbestimmung erfolgt bei Verfahren der 0. Ordnung lediglich auf Basis der Funktionswerte $f(\mathbf{x})$. Verfahren der 1. Ordnung verwenden die ersten Ableitungen der Zielfunktion, Verfahren der 2. Ordnung zusätzlich die berechneten zweiten Ableitungen. Aufgrund der Komplexität und des numerischen Aufwandes zur Bestimmung der zweiten Ableitungen, werden diese Verfahren sehr selten eingesetzt. Durch eine Approximation der zweiten Ableitungen basierend auf den ersten Ableitungen kann der numerische Aufwand reduziert und die Vorteile der Effektivität beibehalten werden (z. B.

beim Quasi-Newton-Verfahren). Die am häufigsten verwendeten Verfahren sind das Gradientenverfahren, das Verfahren der konjugierten Gradienten sowie das Quasi-Newton-Verfahren. Den Suchrichtungsverfahren steht das Simplex-Verfahren gegenüber, welches direkt die Funktionswerte der Zielfunktion verwendet und keine Informationen über die Gradienten benötigt. Dieses Verfahren repräsentiert also ein Verfahren der 0. Ordnung. Es verwendet jedoch keine Suchrichtung, sondern einen n -dimensionalen Körper (Simplex), welcher sich iterativ durch den Lösungsraum in Richtung des Minimums bewegt.

Sollen bei der Optimierung Restriktionen berücksichtigt werden, können diese indirekt oder direkt verarbeitet werden. Bei den indirekten Verfahren wird die ursprüngliche Zielfunktion erweitert und die Restriktion innerhalb der Zielfunktion berücksichtigt. Somit können zur Lösung des Optimierungsproblems die Optimierungsverfahren ohne Restriktionen eingesetzt werden. Die Modifikation der Zielfunktion kann hierbei durch Straffunktionen oder die erweiterte Lagrange-Funktion erfolgen [Har14].

Alternativ zu indirekten Verfahren werden bei direkten Verfahren die Restriktionen direkt vom Optimierungsalgorithmus verarbeitet. Dies erfolgt durch die Definition zulässiger Richtungen oder reduzierter Gradienten gemäß der Restriktionen. Durch die Verwendung reduzierter Gradienten können neben Ungleichheitsrestriktionen auch Gleichheitsrestriktionen verarbeitet werden.

Für eine Bewertung der deterministischen Verfahren werden zunächst die Verfahren der eindimensionalen Optimierung (Liniensuche) betrachtet. Hierzu werden die folgenden Kriterien verwendet [Cav13]:

- Einfachheit: Die Einfachheit des Verfahrens besagt, dass die zugrunde liegende Mathematik einfach und das Verfahren einfach zu implementieren ist.
- Zuverlässigkeit: Die Zuverlässigkeit sagt aus, dass eine Abweichung von der erwarteten Funktionsweise des Verfahrens als unwahrscheinlich angesehen wird und mit hoher Wahrscheinlichkeit eine Lösung erzielt wird.
- Effizienz: Das Optimierungsverfahren konvergiert schnell und benötigt eine geringe Anzahl an Evaluationen. Im Vergleich zur generellen Definition der Effizienz von Optimierungen in dieser Arbeit wird hierbei nicht der mögliche Parallelisierungsgrad bezogen auf die zur Verfügung stehenden Ressourcen berücksichtigt (s. Abschnitt 2.1).

Wie in Tabelle 2.3 zu sehen ist, sind die gradientenbasierte Liniensuche und die gradientenfreie Liniensuche gleichermaßen zuverlässig. Unterschiede zeigen sich in der Einfachheit und in der Effizienz der Verfahren. Aufgrund der Bestimmung bzw. der Approximation der Ableitungen der Zielfunktion ist die gradientenbasierte Liniensuche deutlich komplexer aber auch effizienter als die gradientenfreie Liniensuche, insbesondere bei numerisch aufwendigen Evaluationen der Designvariablen. Eine Sonderrolle nimmt der Trust-Region-Ansatz ein, welcher auf der lokalen Approximation der Zielfunktion durch eine quadratische Funktion basiert. Bei diesem Verfahren kann nicht die gleiche Bewertung wie bei den Verfahren mit Liniensuche erzielt werden [Cav13].

Tabelle 2.3: Bewertung eindimensionaler Optimierungsverfahren, nach [Cav13]

Optimierungsansatz	Einfachheit	Zuverlässigkeit	Effizienz
Gradientenbasierte Liniensuche	○	++	++
Gradientenfreie Liniensuche	++	++	○
Trust-Region	–	○	--

++ sehr hoch; + hoch; ○ neutral; – gering; -- sehr gering

Bei der Bewertung und dem Vergleich der Optimierungsverfahren ohne Restriktionen in Tabelle 2.4 zeigt sich, dass das durch Nelder und Mead [NM65] weiterentwickelte Simplex-Verfahren trotz seiner Einfachheit eine sehr hohe Zuverlässigkeit bei gleichzeitig hoher Effizienz besitzt. Im Gegensatz zum ursprünglichen durch Spendley [SHH62] entwickelten Verfahren konnte die Effizienz des Simplex-Verfahrens deutlich gesteigert werden. Obwohl keine Ableitungen der Zielfunktion verwendet werden, gelten beide Simplex-Verfahren als sehr zuverlässig und effektiv in der Ermittlung eines Optimums.

Tabelle 2.4: Bewertung deterministischer Optimierungsverfahren ohne Restriktionen, nach [Cav13]

Optimierungsverfahren	Einfachheit	Zuverlässigkeit	Effizienz
Simplex nach Spendley	+	++	--
Simplex nach Nelder und Mead	+	++	+
Gradientenverfahren	++	–	–
Konjugierte Gradienten	–	○	○
Newton	++	--	–
DFP-Quasi-Newton	--	○	+
BFGS-Quasi-Newton	--	++	++
Levenberg-Marquardt	–	–	○

++ sehr hoch; + hoch; ○ neutral; – gering; -- sehr gering

Gradientenverfahren bzw. Verfahren des steilsten Abstiegs und die Verwendung konjugierter Gradienten sind einfach in ihrer Anwendung und gewährleisten bei mittlerer Effizienz nur eine mittlere Zuverlässigkeit.

Das Newton-Verfahren kann als Ursprung der gradientenbasierten Optimierungsverfahren gesehen werden und ist daher verglichen mit anderen Verfahren auch einfach in der Anwendung. Hierbei wird die zu minimierende Zielfunktion lokal durch eine quadratische Funktion approximiert und über die Nullstelle der ersten Ableitung der Approximationsfunktion deren Minimum errechnet. Dazu wird jedoch der Wert der zweiten Ableitung der Zielfunktion benötigt, was sehr rechenintensiv ist [BC11].

Eine höhere Zuverlässigkeit und Effizienz werden durch das Quasi-Newton-Verfahren möglich. Hierbei wird die Ermittlung der zweiten Ableitungen durch den schrittweisen Aufbau der Hesse-Matrix ersetzt, was während der eigentlichen Optimierung stattfindet [Har14].

Quasi-Newton-Verfahren stellen das beste Verfahren zur Lösung konventioneller Optimierungsprobleme ohne Restriktionen dar und sind anderen Verfahren vorzuziehen. Die bekanntesten Quasi-Newton-Verfahren sind das Verfahren nach Davidon, Fletcher und Powell (DFP) [Dav59], [FP63], welches eine sehr genaue Liniensuche erfordert, und das Verfahren nach Broyden, Fletcher, Goldfarb und Shanno (BFGS) [Bro70], [Fle70], [Gol70], [Sha70], bei dem auch weniger genaue gradientenbasierte Liniensuchverfahren verwendet werden können. Das BFGS-Verfahren führt daher zu sehr guten Ergebnissen bei gleichzeitig sehr hoher Effizienz [Cav13].

Das Levenberg–Marquardt-Verfahren basiert ebenfalls auf dem Newton-Verfahren in Verbindung mit dem Gradientenverfahren. Konvergenz wird bei diesem Verfahren durch die Verwendung begrenzter Schrittweiten erreicht, wobei analog zum Newton-Verfahren die zweiten Ableitungen der Zielfunktion benötigt werden. Um eine angemessene Effizienz bei der Lösung eines Problems zu erzielen, sind viele Anpassungen nötig, wodurch das Verfahren in der Praxis selten Anwendung findet und gegenüber den Quasi-Newton-Verfahren keine Vorteile bietet [Cav13].

Nach Cavazzuti [Cav13] sollten zur Bearbeitung von Optimierungsproblemen durch deterministischer Optimierungsverfahren ohne Restriktionen ausschließlich Quasi-Newton-Verfahren, insbesondere das BFGS-Quasi-Newton-Verfahren, oder das Simplex-Verfahren nach Nelder und Mead verwendet werden.

Moderne deterministische Optimierungsalgorithmen vereinen zur Steigerung der Zuverlässigkeit und der Effizienz mehrere Optimierungsverfahren. So basiert der von Schittkowski [Sch86], [Sch01a] entwickelte Algorithmus NLPQLP auf Verfahren der nichtlinearen Programmierung (NLP), wobei Teilprobleme der Optimierungsaufgabe durch quadratische Approximationen (Q) der Lagrange-Funktion mit linearisierten Restriktionen (L) gelöst werden. Nach der Lösung der Teilprobleme erfolgt eine Liniensuche unter Verwendung zwei alternativer Zielfunktionen. Dabei werden Straffunktionen und das BFGS-Verfahren verwendet. Zudem unterstützt der Algorithmus die parallele Verarbeitung der beschriebenen Teilschritte (P). Dieser aufgrund seiner hohen Zuverlässigkeit und Effizienz als Stand der Technik eingestufte Algorithmus setzt einen harmonischen Verlauf der Zielfunktion voraus [Cav13]. Da diese im Vorfeld einer Optimierung nicht bekannt ist, kann auch hierbei nicht eindeutig bestimmt werden, ob das globale Optimum eines Problems gefunden wird.

2.6.2 Stochastische Optimierungsverfahren

Im Gegensatz zu deterministischen Optimierungsverfahren, mit denen sehr effizient lokale Optima eines Problems gefunden werden können, liefern stochastische Optimierungsverfahren eine hohe Wahrscheinlichkeit, das globale Optimum eines Problems zu finden. Aufgrund eines implementierten Zufallsanteils sind die Verfahren in der Lage, lokale Optima zu überspringen und das globale Optimum zu ermitteln. Diese oft auch als globale Verfahren bezeichneten Optimierungsverfahren finden daher ein breites Anwendungsfeld in der Praxis [Cav13], [Sch13].

Bei der Entwicklung dieser Verfahren, welche zum Ziel hat, die Nachteile deterministischer Verfahren zu umgehen, haben sich zwei grundlegende Ansätze herausgestellt: die Implementierung stochastischer Anteile in deterministische Verfahren und die Entwicklung eigenständiger stochastischer Verfahren [LRL03].

Beispielhaft für die Implementierung stochastischer Anteile in deterministische Verfahren ist die Verwendung mehrerer Startpunkte. Bei dieser Multistartstrategie werden durch einen meist gleichverteilten Zufallsgenerator oder durch zufallsbasierte Versuchspläne (s. Abschnitt 2.7.1) mehrere Startpunkte für ein deterministisches Optimierungsverfahren generiert, wodurch die Wahrscheinlichkeit erhöht wird, dass eines der gefundenen lokalen Optima auch das globale Optimum ist. Liegen bereits Informationen über den Lösungsraum vor, ist eine gezielte Platzierung der Startpunkte durch den Anwender oft effizienter [Har14]. Da die Verwendung mehrerer Startpunkte kein eigenständiges Optimierungsverfahren, sondern eher eine Kombination vorhandener Optimierungsverfahren darstellt, wird dieses Verfahren auch oft den hybriden und weiteren Verfahren zugeordnet [Sch13].

In erster Linie beschreiben stochastische Optimierungsverfahren eigenständige Verfahren, bei denen die stochastischen Anteile grundlegend im Verfahren implementiert sind. Diese Verfahren stellen den weiteren Inhalt dieses Abschnittes dar. Sie basieren oft auf Analogien, meist aus der Natur, sowie unkonventionellen Ansätzen und verwenden keine mathematische Formulierung der Suchrichtung und keine eindimensionale Optimierung. Somit besteht keine Beschränkung auf konvexe Optimierungsprobleme. In der mathematischen Optimierung wird daher auch der Begriff der Suchstrategien anstatt Optimierungsalgorithmen verwendet. Insbesondere wenn die Einflüsse der einzelnen Designvariablen (Sensitivitäten) nicht analytisch ermittelt werden können, z. B. durch Gradienten der Zielfunktion (s. Abschnitt 2.6.1), zeigen stochastische Optimierungsverfahren deutliche Vorteile gegenüber deterministischen. Restriktionen können bei diesen Verfahren nicht direkt berücksichtigt werden, sondern müssen der Zielfunktion indirekt als Straffunktionen hinzugefügt werden (s. Abschnitt 2.8.1.3) [Sch13].

Aufgrund des Zufallsanteils ist die Anwendung stochastischer Optimierungsverfahren mit einem erhöhten Rechenaufwand verbunden. Im Vergleich zu effizienten deterministischen Algorithmen liegt die Anzahl der benötigten Evaluationen (bzw. Iterationen) oft über dem Faktor 100. Je nach Funktionsweise arbeiten stochastische Verfahren mit teilweise voneinander unabhängigen Evaluationen, welche auf verschiedene Arten parallel bearbeitet werden können, wodurch die Effizienz dieser Verfahren gesteigert wird (s. Kapitel 4).

Es existieren viele verschiedene stochastische Optimierungsverfahren, welche oft auch sehr spezialisierte Lösungen darstellen und auf die jeweilige Problemstellung angepasst sind. Die wichtigsten stochastischen Optimierungsverfahren sind:

- Evolutionäre und genetische Algorithmen [Hol75],[Gol89], Evolutionsstrategien [Rec73]
- Partikelschwarmverfahren (Particle Swarm) [KG06]
- Simuliertes Ausglühen (Simulated Annealing) [KGV83]
- Ameisenkolonie-Algorithmus (Ant Colony Optimization, ACO) [CDM92]
- Intelligente Wassertropfen (Intelligent Water Drops, IWD) [SH07]
- Künstliche Immunsysteme (Artificial Immune System, AIS) [Kep94]
- Stochastische Verbreitungssuche (Stochastic Diffusion Search, SDS) [Bis89]
- Spieltheorie (Game Theory) [Nas51]
- Farnoptimierung [SPS⁺06], [SG15]

In der Produktentwicklung haben sich vor allem evolutionäre und genetische Algorithmen, Partikelschwärme und das simulierte Abkühlen etabliert. Diese Verfahren werden im Folgenden näher beschrieben. Aufgrund der besonderen Eignung genetischer Algorithmen zur Lösung diskreter und kombinatorischer Optimierungsprobleme, wie sie in dieser Arbeit bei der Optimierung der Prozessprioritäten zu finden sind (s. Abschnitt 6.4.2), werden die Struktur und die Anwendung genetischer Algorithmen detaillierter erläutert.

2.6.2.1 Evolutionäre und genetische Algorithmen

Evolutionäre Algorithmen bilden die wichtigste Gruppe stochastischer Optimierungsverfahren. Sie basieren auf Analogien zur biologischen Evolution und stellen eine Teilmenge der biologisch inspirierten Programmierung dar. Im Optimierungsprozess wird der Wettbewerb der am besten an die Umgebung angepassten Individuen nachgebildet (*survival of the fittest*). Die Eigenschaften der Individuen werden dabei durch Gene charakterisiert. Zielführende Gene werden identifiziert und von den Eltern an die Nachkommen weitergegeben, bis sich in einem iterativen Prozess über mehrere Generationen die besten Gene durchgesetzt haben und somit das Optimum gefunden wurde.

Innerhalb der evolutionären Algorithmen werden in erster Linie Evolutionsstrategien und genetische Algorithmen unterschieden. Diese Differenzierung basiert primär auf der unterschiedlichen Repräsentation der Gene, wodurch sich jedoch auch weitere Unterschiede in der Umsetzung sowie den Operatoren der Verfahren ergeben [Har14], [SG15].

Bei *Evolutionsstrategien* werden die Gene durch reelle Zahlen repräsentiert, welche in der Regel die Designvariablen darstellen. Die Operatoren werden somit direkt auf den Designvariablen angewendet.

Genetische Algorithmen hingegen verwenden eine Codierung der Gene, nicht die reellen Werte der Designvariablen selbst. Die Werte des Designvariablenvektors \mathbf{x} werden dabei oft durch binäre Zahlen repräsentiert, wie beispielhaft in der folgenden Gleichung dargestellt ist [Har14].

$$\mathbf{x} = \begin{bmatrix} (11011\dots01), \\ (01101\dots00), \\ \dots \end{bmatrix} \quad (2.10)$$

Die binäre Codierung der Designvariablen ist vor allem bei diskreten Optimierungsproblemen von Vorteil. Jedoch führt die reine Verwendung der Binärcodierung aufgrund der unterschiedlichen Gewichtung der binären Werte zu dem Problem, dass kleine Änderungen im Binärcode in großen Änderungen der reellen Zahlenwerte resultieren können, was insbesondere bei Mutation zu Problemen führen kann, wie im weiteren Verlauf dieses Abschnittes noch näher erläutert wird. Eine Alternative zur konventionellen Binärcodierung ist die Cray-Codierung, welche sicherstellt, dass durch die Invertierung eines Bits die codierte Zahl um 1 erhöht oder reduziert [Har14]. Der Vergleich zwischen der Binärcodierung und der Cray-Codierung ist detailliert in Tabelle A.1 im Anhang A.2 dargestellt.

Ja nach Problemstellung existiert eine große Vielzahl unterschiedlicher evolutionärer und genetischer Algorithmen. Im Folgenden werden daher in erster Linie die grundlegende Funktionsweise und die Anwendung beschrieben. Trotz der großen Vielzahl basieren die meisten Algorithmen auf der gleichen Struktur, welche in Abbildung 2.22 dargestellt ist.

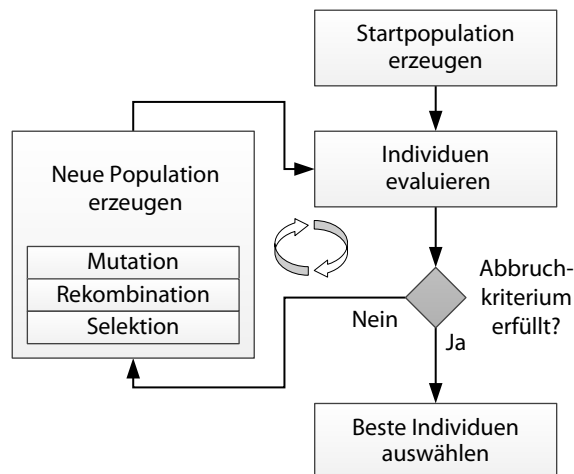


Abbildung 2.22: Grundstruktur eines einfachen evolutionären Algorithmus, nach [Poh00]

Die Ausgangsbasis bildet in der Regel eine zufällig erzeugte Startpopulation, deren Individuen zunächst evaluiert werden. Jedes Individuum stellt dabei eine Variante der Optimierung dar, welche durch die Designvariablen repräsentiert wird. Nach Prüfung eines oder mehrerer Abbruchkriterien erfolgt die Erzeugung einer neuen Population aus der Startpopulation. Dabei kommen verschiedene evolutionäre Operatoren zum Einsatz, z. B. Selektion, Rekombination und Mutation. Da die Individuen der erzeugten Population die Nachkommen der vorigen Population darstellen, werden die unterschiedlichen Populationen auch als Generationen bezeichnet. Nach der Erzeugung einer Generation werden die Individuen der Generation evaluiert. Dieser iterative Prozess wird solange wiederholt, bis ein Abbruchkriterium erfüllt ist und die Optimierung beendet wird. Anschließend werden durch den Anwender auf Basis der Zielkriterien das oder die besten Individuen ausgewählt.

Die Auswahl eines Individuums für die Erzeugung der Nachkommen wird als *Selektion* bezeichnet. Diese erfolgt auf Basis eines Zahlenwertes, der *Fitness* f_{Fit} , welcher zunächst jedem Individuum zugeordnet wird. Die Fitness eines Individuums stellt dessen Fortpflanzungswahrscheinlichkeit bzw. Selektionswahrscheinlichkeit dar. Sie wird aus dem Zielfunktionswert $f(\mathbf{x})$ des Individuums und den Zielfunktionswerten aller anderen für die Selektion zur Verfügung stehenden Individuen (Selektionspool) mittels der Fitnessfunktion berechnet. Bessere Individuen, welche einen besseren Zielfunktionswert besitzen, erhalten eine höhere Fitness als schlechtere Individuen. Die Fitness ist somit ein Maß dafür, wie gut ein Individuum die Zielkriterien erfüllt und wie gut es an seine Umgebung angepasst ist. Im Gegensatz zum Zielfunktionswert, welcher nur von den Designvariablen des Individuums abhängt, ist die Fitness eines Individuums immer im Vergleich zu den anderen Individuen des Selektionspools zu betrachten. Da die Fitness eines Individuums dessen Fortpflanzungswahrscheinlichkeit repräsentiert und negative Wahrscheinlichkeiten nicht existieren, muss bei der Fitnesszuweisung zudem sichergestellt werden, dass nur positive Fitnesswerte vorliegen [Poh00].

Zur Fitnesszuweisung haben sich folgende Verfahren etabliert [Poh00]:

- Proportionale Fitnesszuweisung [Gol89]
- Rangbasierte Fitnesszuweisung (Ranking) [Bak85]
- Multikriterielle Fitnesszuweisung [ZDT00]

Die *proportionale Fitnesszuweisung* stellt das einfachste Verfahren dar. Dabei ist der dem Individuum zugewiesene Fitnesswert proportional zu dem Zielfunktionswert des Individuums ($f_{Fit} \sim f(\mathbf{x})$). Es erfolgt also eine Skalierung des Zielfunktionswertes, welche linear, linear dynamisch (über die Generationen veränderlich), logarithmisch oder exponentiell sein kann [Poh00]. Der Nachteil der proportionalen Fitnesszuweisung ist das Auftreten von Dominanz. Besitzt ein Individuum einen sehr guten Zielfunktionswert gegenüber den anderen Individuen, führt dies auch zu einem sehr hohen Fitnesswert gegenüber den anderen Individuen, was in einer hohen Selektionswahrscheinlichkeit resultiert und dazu führt, dass dieses Individuum wiederholt selektiert wird. Aufgrund der sehr kleinen Selektionswahrscheinlichkeit der übrigen Individuen werden diese kaum oder gar nicht selektiert. Ein Individuum dominiert somit alle anderen Individuen und der Algorithmus erreicht vorzeitige Konvergenz [KBB⁺15].

Das Auftreten von Dominanz kann durch eine *rangbasierte Fitnesszuweisung* vermieden werden. Bei diesem Verfahren werden die Individuen des Selektionspools zunächst gemäß ihres Zielfunktionswertes sortiert. Die Zuweisung der Fitness erfolgt dann basierend auf der Position bzw. dem Rang eines Individuums. Somit wird eine gleichmäßige Skalierung der Fitnesswerte über den gesamten Selektionspool gewährleistet. Die Fitnesszuweisung kann hierbei linearen oder nichtlinearen Charakter haben. Eine lineare Fitnesszuweisung (lineares Ranking) wird durch die folgende Fitnessfunktion erzielt:

$$f_{Fit}(r_I) = 2 - p_s + 2 \cdot (p_s - 1) \cdot \frac{n_I - r_I}{n_I - 1} \quad \text{mit} \quad 1 \leq p_s \leq 2 \quad (2.11)$$

Die Fitness eines Individuums I gegenüber den anderen Individuen hängt ausschließlich von dessen Rang r_I ab. Weiterhin werden für die Berechnung die Größe des Selektionspools n_I (entspricht in der Regel der Populationsgröße) sowie der Selektionsdruck p_s verwendet. Der Selektionsdruck ist ein Maß für die Wahrscheinlichkeit der Selektion des besten Individuums, verglichen mit der durchschnittlichen Selektionswahrscheinlichkeit aller Individuen des Selektionspools. Da bei der Fitnesszuweisung keine negativen Fitnesswerte vorgesehen sind, ist der Selektionsdruck auf den Wertebereich $[1; 2]$ begrenzt [Poh00]. Wie in Abbildung 2.23 dargestellt, wird durch den Selektionsdruck der Gradient der Fitnessfunktion beim linearen Ranking gesteuert.

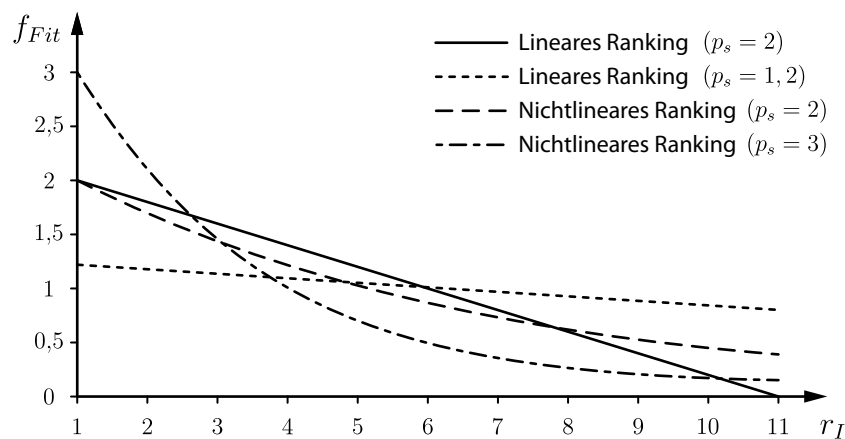


Abbildung 2.23: Fitnesszuweisung bei linearem und nichtlinearem Ranking, nach [Poh00]

Je größer der Gradient der Fitnessfunktion, desto höher ist die Fitness des besten Individuums und somit dessen Selektionswahrscheinlichkeit. Dieser Effekt kann durch die Verwendung einer nichtlinearen Fitnessfunktion verstärkt werden (nichtlineares Ranking). Dabei sind auch höhere Werte für den Selektionsdruck möglich. Für die mathematische Formulierung nichtlinearen Rankings wird auf [Poh00] verwiesen.

Da Optimierungsprobleme oft nicht nur auf ein Optimierungsziel beschränkt sind, sondern mehrere Zielkriterien verfolgt werden, hat sich neben den bisher beschriebenen monokriteriellen Verfahren die *multikriterielle Fitnesszuweisung* etabliert. Hierbei ist der Rang r_I eines Individuums in einer Population davon abhängig, wie viele Individuen $n_{I,dom}$ dieses Individuum dominieren, also in einem der Zielkriterien einen besseren Wert liefern, ohne in einem anderen schlechter zu sein [Fon95]. Der Rang r_I eines Individuums berechnet sich somit nach:

$$r_I = 1 + n_{I,dom} \quad (2.12)$$

Dadurch wird sichergestellt, dass alle Individuen der Pareto-optimalen Menge P (s. Abschnitt 2.4) den höchsten Rang von 1 erhalten und demnach gleichwertig sind. Dieses Vorgehen stellt eine Möglichkeit der Bearbeitung multikriterieller Optimierungsaufgaben dar. Dominanz, Pareto-Optimalität und weitere Strategien der multikriteriellen Optimierung werden in Abschnitt 2.8.1 beschrieben.

Nach der Fitnesszuweisung erfolgt die *Selektion* der Individuen. Dabei wird meistens eines der folgenden Verfahren angewendet [Poh00]:

- Rouletteselektion (Roulette-Wheel Selection) [Bak87]
- Stochastisch universelle Selektion (Stochastic Universal Sampling) [Bak87]
- Turniers Selektion (Tournament Selection) [BT95]
- Truncation-Selektion (Abschneideselektion) [MSV95]

Die Selektionswahrscheinlichkeit eines Individuums resultiert aus dessen Fitnesswert, welcher zunächst auf den gesamten Selektionspool normiert wird. Tabelle 2.5 zeigt beispielhaft die Fitnesszuweisung durch lineares Ranking und die aus den Fitnesswerten abgeleitete normierte Fitness $f_{Fit,norm}$ für eine Populationsgröße von $n_I = 5$. Dabei ist zu erkennen, dass die Summe der Fitnesswerte der Populationsgröße entspricht und die Summe der normierten Fitnesswerte gleich 1 ist.

Tabelle 2.5: Lineares Ranking und Normierung des Fitnesswertes

Individuum	Rang r_I	Fitness f_{Fit}^*	Norm. Fitness $f_{Fit,norm}$
I_1	1	1,8	0,36
I_2	2	1,4	0,28
I_3	3	1,0	0,20
I_4	4	0,6	0,12
I_5	5	0,2	0,04
Summe:		5	1

* Lineares Ranking ($p_s = 1,8$)

Das bekannteste Selektionsverfahren ist die *Rouletteselektion*, bei der den Individuen gemäß ihres absoluten Fitnesswertes Bereiche auf einer Linie zugeordnet werden. Die Größe eines Bereiches entspricht der jeweiligen Fitness bzw. der normierten Fitness. Auf Basis einer generierten Zufallszahl, welche zwischen 0 und der Länge der Linie bzw. zwischen 0 und 1 liegt, erfolgt die Auswahl des Individuums. Das Individuum, in dessen Bereich die Zufallszahl liegt, wird selektiert. Dieser Vorgang wird so oft wiederholt, bis die Anzahl der zu selektierenden Individuen erreicht ist. Es wird also das zufällige Drehen eines Rouletterades nachgebildet. Bereits selektierte Individuen werden nicht von der weiteren Selektion ausgeschlossen. Das Verfahren wird daher auch als *stochastic sampling with replacement* bezeichnet [Poh00].

Da die Selektion gemäß der absoluten Fitnesswerte, also fitnessproportional erfolgt und bereits selektierte Individuen mit der gleichen Wahrscheinlichkeit wieder selektiert werden können, besteht auch hierbei die Gefahr der Dominanz und somit der vorzeitigen Konvergenz, wenn ein Individuum aufgrund seines großen Bereiches mehrfach selektiert wird. Abbildung 2.24 zeigt die Rouletteselektion am Beispiel der in Tabelle 2.5 aufgeführten normierten Fitnesswerte. Die mögliche Position der Zufallszahl ist durch Zeiger dargestellt. Aufgrund des großen Bereichs von I_1 besteht eine große Wahrscheinlichkeit, dass dieses Individuum mehrfach selektiert wird und die restlichen Individuen dominiert.

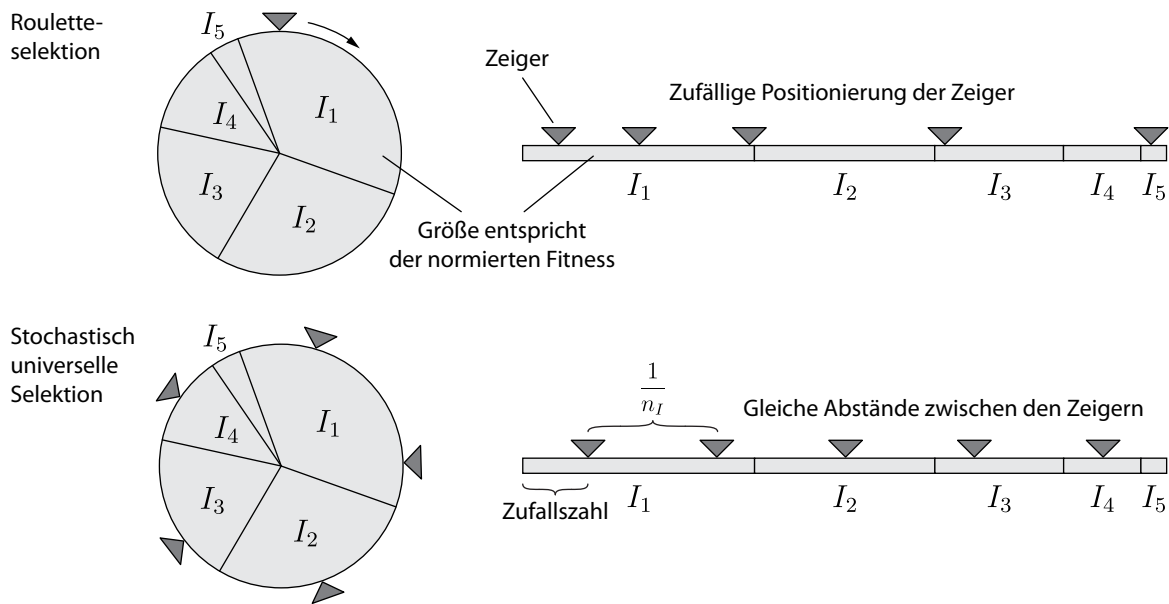


Abbildung 2.24: Rouletteselektion und stochastisch universelle Selektion, nach [Poh00], [GKK04], [KBB⁺15]

Bei der *stochastisch universellen Selektion* werden die Individuen ebenfalls fitnessproportional ausgewählt. Analog zur Rouletteselektion werden den Individuen Bereiche einer Linie zugeordnet. Die zur Auswahl der Individuen verwendeten Zeiger werden bei diesem Verfahren jedoch gleichmäßig auf der Linie im Abstand von $1/n_I$ verteilt. Die Generierung der Zufallszahl erfolgt einmal im Selektionsprozess. Durch die Zufallszahl, welche Werte zwischen 0 und $1/n_I$ annehmen kann, werden alle Zeiger auf der Linie verschoben und anschließend die Individuen selektiert, welche durch die Zeiger markiert sind. Die fitnessproportionale Selektion der Individuen ist sehr schnell und einfach durchführbar. Zur Reduzierung von Dominanz sollte die stochastisch universelle Selektion der Rouletteselektion vorgezogen werden [Poh00].

Die *Turnierselektion* beinhaltet die Selektion der Individuen aus Turnieren. Die Auswahl der Individuen für ein Turnier erfolgt gleichverteilt zufällig aus dem Selektionspool, wobei die Anzahl der gewählten Individuen der zuvor festgelegten Turniergröße entspricht. In dem Turnier wird dann das Individuum mit der höchsten Fitness selektiert. Dabei wird ausschließlich die Rangfolge der Fitnesswerte der Individuen verwendet. Je größer also die festgelegte Turniergröße, desto kleiner ist die Wahrscheinlichkeit auch schwache Individuen zu selektieren. Die Anzahl der Turniere entspricht der Anzahl der zu selektierenden Individuen. Alle Turniere können unabhängig voneinander und somit parallel durchgeführt werden [Poh00].

Im Gegensatz zu den bisher beschriebenen Selektionsverfahren, welche die natürliche Selektion nachbilden, werden bei der *Truncation-Selektion* nur die Individuen selektiert, deren Fitnesswert oberhalb eines definierten Schwellenwertes liegt. Da diese Art der Selektion nicht aus der biologischen Evolution abgeleitet ist, wird auch von einem künstlichen Selektionsverfahren gesprochen. Die Fitnesszuweisung erfolgt ausschließlich rangbasiert. Dieses Selektionsverfahren wird vor allem bei sehr großen Populationen angewendet [MSV95].

Weiterhin wird bei den beschriebenen Selektionsverfahren aufgrund des stochastischen Anteils nicht sichergestellt, dass das Individuum mit der höchsten Fitness für die Gene-

rierung der Nachkommen selektiert wird. Daher wird oft zusätzlich *Elitismus* verwendet, wodurch eine festgelegte Anzahl der besten Individuen direkt in die Folgegeneration kopiert wird. Es findet also eine Replikation der Individuen statt. Dieses Verfahren beugt zudem einer möglichen Verschlechterung in der Folgegeneration vor [KBB⁺15].

Nach der Selektion werden aus den selektierten Individuen die Nachkommen erstellt. Dies erfolgt durch *Rekombination*, bei der die Informationen der selektierten Individuen miteinander kombiniert werden. Je nach der Repräsentation der Designvariablen in den Individuen haben sich bei der Parameteroptimierung folgende Rekombinationsverfahren etabliert [Poh00]:

- Alle Repräsentationen der Designvariablen (reell, ganzzahlig, binär):
 - Diskrete Rekombination (Discrete Recombination) [MSV93] (reelle Variablen), entspricht dem Uniform Crossover [Sys89] (binäre Variablen)
- Reelle Designvariablen:
 - Intermediäre Rekombination (Intermediate Recombination) [MSV93]
 - Linien-Rekombination (Line Recombination) [MSV93]
 - Erweiterte Linien-Rekombination (Extended Line Recombination) [Müh94]
- Binäre Designvariablen:
 - Single-Point-, Double-Point-, Multi-Point Crossover
 - Shuffle Crossover [CES89]
 - Crossover with Reduced Surrogate [Boo87]

Durch die diskrete Rekombination können alle Arten von Designvariablen verarbeitet werden. Bei der Verwendung binärer Repräsentationen der Designvariablen, welche durch die binäre Codierung genetischer Algorithmen entstehen, wird auch der Begriff *Crossover* (dt. Kreuzung) verwendet. Im Gegensatz zum Single-Point-, Double-Point- oder Multi-Point Crossover, bei denen das Chromosom eines Individuums nur an einer bzw. bestimmten Stellen für den Austausch aufgetrennt wird, ist bei der diskreten Rekombination bzw. dem Uniform Crossover ein freier Austausch aller Variablenwerte möglich. Aufgrund der Analogie zur biologischen Evolution werden die Werte der Designvariablen auch als Allele bezeichnet. Das Single-Point-Crossover anhand eines aus zehn Genen bestehenden Chromosoms ist in Abbildung 2.25 dargestellt.



Abbildung 2.25: Funktionsweise der genetischen Operatoren Rekombination und Mutation, nach [GKK04]

Bei der Bearbeitung kombinatorischer Optimierungsprobleme werden spezielle Rekombinationsverfahren verwendet (Edge Recombination, Maximal Preservative Crossover oder

Order bzw. Position Crossover). Für eine detaillierte Beschreibung der Rekombinationsverfahren wird auf [Poh00] verwiesen.

Nach der Rekombination erfolgt die *Mutation* der erzeugten Individuen. Dabei werden die Variablenwerte durch kleine Störungen verändert. Diese Mutationen finden in der Regel mit einer festgelegten Wahrscheinlichkeit statt, der Mutationsrate, welche oft relativ klein gehalten ist (z. B. 10 %), da sonst das durch die Rekombination erzeugte gute Genmaterial wieder zerstört wird. Mutation ist insbesondere bei kleinen Populationen sinnvoll, um das Auftreten vorzeitiger Konvergenz zu verhindern und lokale Optima zu umgehen, da bei reiner Rekombination das genetische Material ausschließlich zwischen den Individuen getauscht wird und kein neues genetisches Material in die Population eingebracht wird. Aufgrund der genetischen Vielfalt großer Populationen ist hierbei der evolutionäre Operator der Mutation nicht so wichtig wie bei kleinen Populationen [GKK04]. Analog zur Rekombination werden auch bei der Mutation verschiedene Verfahren in Abhängigkeit von der Repräsentation der Designvariablen unterschieden [Poh00]:

- Reelle Designvariablen:
 - Kontinuierliche und verteilte Mutation [VMC95], [SVM96]
 - Mutation einer Evolutionsstrategie [OGH94], [HOG95]
- Ganzzahlige Designvariablen:
 - Analog zur reellen Mutation (für Parameteroptimierung)
- Binäre Designvariablen:
 - Veränderung des Variablenwertes
- Reihenfolgeoptimierung:
 - Vertauschung von Variablen (Swap-/Insert-/Reverse-/Scramble-Mutation)

Wie in Abbildung 2.25 dargestellt, ist die Mutation binärer Variablen sehr einfach, da nur zwei Zustände möglich sind und somit lediglich der Zahlenwert (0 oder 1) geändert wird. Dabei ist zu beachten, dass bei der Verwendung der konventionellen Binärcodierung eine Veränderung eines Gens bereits zu einer großen Änderung des Designvariablenwertes führen kann, welche ggf. außerhalb der oberen und unteren Grenzen der Designvariablen liegen können. Durch eine Verwendung der Cray-Codierung wird sichergestellt, dass durch die Invertierung eines Bits die codierte Zahl nur um 1 erhöht oder reduziert wird [Har14]. Alternativ müssen geeignete Reparaturmechanismen verwendet werden, um sicherzustellen, dass die mutierten Allele auch in zulässigen Werten der Designvariablen resultieren [GKK04]. Eine detaillierte Beschreibung der Mutationsverfahren ist in [Poh00] zu finden.

Nach der Erzeugung der Nachkommen aus den selektierten Individuen durch Rekombination und Mutation wird die neue Population gebildet. Oft wird die ursprüngliche Generation vollständig durch die erzeugten Nachkommen ersetzt. Es können aber auch nur bestimmte Individuen der ursprünglichen Generation ersetzt und auch nur bestimmte Nachkommen der Population hinzugefügt werden. Dieser Prozess wird als *Wiedereinfügen* bzw. *Reinsertion* bezeichnet. Beim *einfachen Wiedereinfügen* (Pure Reinsertion) entspricht die Zahl der Nachkommen der Populationsgröße und alle Nachkommen werden in die Population übernommen. *Zufälliges* bzw. *elitäres Wiedereinfügen* (Uniform

bzw. Elitest Reinsertion) beschreibt das Einfügen einer kleineren Anzahl von Nachkommen als Individuen in der Population. Dabei werden die Individuen der Population durch die Nachkommen gleich verteilt zufällig ersetzt (Uniform Reinsertion) oder es werden ausschließlich die schlechtesten Individuen der Population ersetzt (Elitest Reinsertion). Alternativ kann das Wiedereinfügen auch durch eine Vorabauswahl der Nachkommen erfolgen (Reinsertion with Offspring Selection), wobei nur die besten Nachkommen (z. B. durch Truncation-Selektion) in die Population eingefügt werden [Poh00].

Einen Spezialfall des Wiedereinfügens stellt die *Steady-State*-Reproduktion dar, welche auch oft den Selektionsverfahren zugeordnet wird [Poh00], [GKK04]. Die Anzahl der erzeugten Individuen ist hierbei geringer als die aktuelle Populationsgröße. Die erzeugten Individuen werden anschließend der vorhandenen Population hinzugefügt und ersetzen dabei die schlechtesten Individuen, was zur Folge hat, dass die reproduzierten Individuen mindestens nicht schlechter, in der Regel jedoch immer besser werden.

Die Erzeugung und Bewertung neuer Individuen wird solange fortgesetzt, bis ein *Abbruchkriterium* erfüllt ist und der Algorithmus endet. Da bei der anschließenden Auswahl des bzw. der besten Individuen nicht genau festgestellt werden kann, ob dies ein lokales oder das globale Optimum darstellt, werden zur Erhöhung der Wahrscheinlichkeit das globale Optimum zu finden, mehrere Abbruchkriterien verwendet. Hierbei wird zwischen direkten und indirekten Abbruchkriterien unterschieden. *Direkte Abbruchkriterien* beinhalten die Vorgabe einer maximalen Anzahl von Generationen oder Evaluationen, einer maximalen Laufzeit oder der Vorgabe eines zu erreichenden Zielfunktionswertes, welcher das Optimum darstellt. Mit Ausnahme der Vorgabe eines Zielfunktionswertes kann nach Beendigung der Optimierung jedoch nicht festgestellt werden, ob das globale Optimum gefunden wurde. Diese Kriterien limitieren daher in erster Linie die maximale Laufzeit des Algorithmus und sind kein Maß für die Güte der erreichten Lösung. Zur Einschätzung der Lösungsgüte werden zusätzlich *abgeleitete Abbruchkriterien* verwendet. Werden diese erreicht, liegt in der Regel eine ausreichende Wahrscheinlichkeit vor, das globale Optimum gefunden zu haben. Folgende Kriterien sind möglich [Poh00]:

- Standardabweichung: Standardabweichung der Zielfunktionswerte der aktuellen Generation
- Laufender Mittelwert: Differenz zwischen dem Mittelwert der besten Zielfunktionswerte der letzten Generationen und dem besten Zielfunktionswert der aktuellen Generation
- Guter schlechtester Wert: Differenz zwischen dem Zielfunktionswert des besten und des schlechtesten Individuums der aktuellen Generation
- Phi: Quotient aus dem Mittelwert aller Zielfunktionswerte und dem besten Zielfunktionswert der aktuellen Generation
- Kappa: Gleichartigkeit der Variablenwerte der Individuen der Population

Neben der Wahl des Abbruchkriteriums hat bei der Verwendung eines EA bzw. GA die Wahl der *Populationsgröße* einen entscheidenden Einfluss auf die Effektivität der Optimierung. Prinzipiell liefert eine größere Population auch eine höhere Wahrscheinlichkeit, das globale Optimum eines Problems zu finden, da generell mehr Individuen und somit ein größerer Genpool existiert. Liegen keine Erfahrungswerte zur Wahl der Populationsgröße vor, kann die folgende Gleichung verwendet werden:

$$n_I \geq 2 \cdot n \cdot n_f ; \quad n_I \geq 16 \quad (2.13)$$

Die Populationsgröße n_I wird somit in Abhängigkeit von der Anzahl der Designvariablen n und der Anzahl der Zielkriterien n_f gewählt. Dabei sollte mindestens eine Populationsgröße von 16 verwendet werden [Cav13].

EA und GA eignen sich vor allem zur Lösung komplexer Probleme, bei un stetigen oder zerklüfteten Zielfunktionen sowie für kombinatorische oder diskrete Optimierungsprobleme [Har14]. Insbesondere GA sind aufgrund der im Folgenden zusammengefassten Vorteile für diese Art von Optimierungsproblemen geeignet [SD08]:

- Da GA kodierte Repräsentationen der Designvariablen anstelle der realen Werte verarbeiten, können verschiedene Parameterarten als Designvariablen fungieren, z. B. Zeichenketten (Strings), kontinuierliche und diskrete Werte oder Entscheidungsbäume.
- GA verwenden Fitnesswerte zur Bestimmung guter Lösungen anstelle von Ableitungen der Zielfunktion, was das Lösen verschiedenartiger Optimierungsprobleme unterstützt, z. B. kontinuierliche oder diskrete Probleme.
- Anstelle eines einzigen Zielfunktionswertes verwenden GA die gesamte Population an Lösungen, wodurch eine hohe Wahrscheinlichkeit gewährleistet wird, das globale Optimum eines Problems zu finden und lokale Optima zu umgehen.
- Durch den implementierten Zufallsanteil eines GA wird ebenfalls die Wahrscheinlichkeit erhöht, aus lokalen Optima zu springen und das globale Optimum zu finden.

Aufgrund dieser Eigenschaften eignen sich genetische Algorithmen neben der konventionellen Parameteroptimierung auch zur Unterstützung der Konzeptentwicklung in den frühen Phasen der Produktentwicklung (s. Abschnitt 3.2.3).

Effektivität und Effizienz evolutionärer und genetischer Algorithmen wurden hinsichtlich unterschiedlicher Anwendungsfälle und Zielstellungen stetig verbessert, wodurch verschiedene Algorithmen entstanden sind. Eine Übersicht bekannter und weitverbreiteter Algorithmen ist in Tabelle A.2 im Anhang A.2 zu finden.

2.6.2.2 Partikelschwarmverfahren (Particle Swarm)

Das von Kennedy und Eberhart [KE95] entwickelte Partikelschwarmverfahren zählt ebenfalls zur biologisch inspirierten Programmierung und dient der Bearbeitung von Optimierungsproblemen mit reellen Designvariablen. Partikelschwärme wurden zunächst als Simulationsmodelle für Sozialverhalten konzipiert und beruhen auf der Modellierung sozialer Interaktionen. Im Gegensatz zu evolutionären Algorithmen werden Verbesserungen nicht durch einen Selektionsprozess, sondern durch die gegenseitige Nachahmung und das Lernen benachbarter Individuen erreicht. Hierbei wird das Schwarmverhalten z. B. von Vögeln oder Fischen hinsichtlich optimaler Futterplätze auf die Lösung von Optimierungsproblemen übertragen.

Jedes Individuum eines Schwarms enthält einen Genotyp, welcher die aktuellen Werte der Designvariablen repräsentiert, sowie zusätzliche Informationen wie den Veränderungsvektor, der zur Modifikation der Individuen dient, und den besten bisher auf dem Weg des Individuums gefundenen Punkten des Designvariablenraumes.

Der Veränderungsvektor des Individuums wird in jeder Generation durch soziale Interaktion mit benachbarten Individuen modifiziert und auf den Genotyp angewendet. Bei

der Modifikation des Vektors wird zum einen das Bestreben des Individuums, zu seinen Erfolgen zurückzukehren, berücksichtigt, zum anderen die Orientierung an den besten Erfolgen der benachbarten Individuen. Die Modifikation wird durch drei Faktoren bestimmt: einen Trägheitsfaktor, einen Faktor, welcher den Einfluss der gespeicherten besten Positionen festlegt, und einen sozialen Faktor, der die Orientierung des Individuums an dessen Nachbarn beeinflusst. Zudem sind im Einfluss der gespeicherten besten Positionen und in der Interaktion mit den benachbarten Individuen Zufallsanteile implementiert, um lokale Optima zu umgehen.

Da ein beliebig großer Veränderungsvektor sehr schnell in einem rein zufälligen Verhalten der Individuen resultiert, wird durch festgelegte Werte der einzelnen Faktoren die Größe des Veränderungsvektors limitiert. Geeignete Werte und eine beispielhafte Umsetzung der beschriebenen Prozesse sind in [Wei15] und [SG15] zu finden.

2.6.2.3 Simuliertes Ausglühen (Simulated Annealing)

Das simulierte Ausglühen basiert ebenfalls auf einer Analogie zur Natur. Bei dem von Kirkpatrick et al. [KGV83] entwickelten Optimierungsverfahren, welches eine Weiterentwicklung des Metropolis–Hastings-Algorithmus [MRR⁺53] darstellt, werden keine biologischen Prozesse, sondern der physikalische Prozess der Kristallisation beim Abkühlen einer Schmelze nachgebildet.

Wird eine Schmelze langsam genug abgekühlt, bildet sich eine perfekte Kristallstruktur aus, welche keine Gitterfehler aufweist und den geringsten möglichen Energiezustand besitzt. Erfolgt der Abkühlvorgang schneller, entstehen in der Kristallstruktur Gitterfehler und das System besitzt einen erhöhten Energiezustand. Bezogen auf den Optimierungsprozess repräsentiert dieser Zustand eine lokales Minimum. Im Gegensatz zum langsamen Abkühlvorgang kann das System aufgrund der reduzierten Temperatur dieses Minimum nicht überwinden, sondern verharrt in diesem Zustand. Die Wahrscheinlichkeit, ein lokales Minimum zu überwinden reduziert sich dabei mit abnehmender Temperatur.

Bei einer Optimierung wird ausgehend von einem Punkt \mathbf{x}_a im Designvariablenraum und einer Starttemperatur zufällig ein neuer Punkt \mathbf{x}_b innerhalb einer festlegten Schrittweite erzeugt, welcher nur akzeptiert wird, wenn er einen besseren Zielfunktionswert $f(\mathbf{x}_b)$ generiert oder bei ausreichender Wahrscheinlichkeit zufällig ausgewählt wird. Die Wahrscheinlichkeit hängt hierbei von der aktuellen Temperatur ab. Somit besteht die Möglichkeit, lokale Optima zu überwinden. Mit zunehmender Anzahl von Iterationen wird die Temperatur abgesenkt und somit die Wahrscheinlichkeit und die Schrittweite reduziert bis das Optimum gefunden wurde [Har14].

Da die Generierung eines neuen Punktes im Designvariablenraum nur von dessen Vorgänger abhängt und immer die Wahrscheinlichkeit besteht, auch schlechtere Zielfunktionswerte zu akzeptieren, besteht die Gefahr, dass während der Optimierung auch aus dem globalen Optimum in ein lokales Optimum gesprungen wird. Am Ende der Optimierung sollten also in jedem Fall noch einmal alle berechneten Zielfunktionswerte überprüft werden.

2.6.2.4 Abschließende Bemerkungen

Stochastische Optimierungsverfahren stellen einen alternativen Ansatz zu deterministischen Verfahren dar, wobei in erster Linie die Zielstellung verfolgt wird, die Nachteile deterministischer Verfahren zu reduzieren, insbesondere in der Ermittlung des globalen Optimums eines Problems. Dabei sind stochastische Verfahren nicht immer effizient, da im Gegensatz zu deterministischen Verfahren eine deutliche größere Anzahl an Evaluationen benötigt wird. Da die Evaluationen in bestimmten Grenzen voneinander unabhängig sind, kann die Effizienz durch verschiedene Parallelisierungsmethoden deutlich gesteigert werden (s. Kapitel 4). Zudem dient die große Anzahl an Evaluationen auch dazu, das Optimierungsproblem und dessen Zusammenhänge besser zu verstehen. Weiterhin liegen die Vorteile stochastischer Verfahren in der Verarbeitung diskreter Designvariablen und kombinatorischer Optimierungsprobleme. Insbesondere evolutionäre und genetische Algorithmen finden hierbei ein breites Anwendungsfeld. Auch wenn deren Anwendung häufig kontrovers diskutiert wird, liefern sie auch bei komplexen Parameterräumen schnell eine große Menge an guten Lösungen für ein Problem. Das als noch relativ neuartig geltende Particle Swarm-Verfahren ist vor allem bei unregelmäßigen Zielfunktionen mit vielen lokalen Optima geeignet und erfordert noch weitere Untersuchungen in der Anwendung [Cav13], [SG15]. Die Charakteristiken und Anwendungsfelder der beschriebenen stochastischen Optimierungsverfahren sind in Tabelle 2.6 zusammengefasst.

Tabelle 2.6: Charakteristik stochastischer Optimierungsverfahren [Cav13]

Optimierungsverfahren	Charakteristik und Anwendung
Evolutionstrategien	Zuverlässiges und robustes mutationsbasiertes Verfahren, Zur Untersuchung des Designvariablenraumes geeignet
Genetische Algorithmen	Zuverlässiges Crossover-basiertes Verfahren, Bei multikriteriellen Problemen den ES vorzuziehen, Zur Untersuchung des Designvariablenraumes sollte der Einfluss der Mutationen erhöht werden
Particle Swarm	Relativ neuartiges Verfahren, was noch genauer untersucht werden muss, Vor allem bei unregelmäßigen Zielfunktionen mit vielen lokalen Optima geeignet
Simulated Annealing	Bei diskreten und kombinatorischen Optimierungsproblemen geeignet, Nicht bei allen Optimierungsproblemen effizient

2.6.3 Hybride Optimierungsverfahren und Weitere

Motiviert durch die Verwendung der spezifischen Vorteile unterschiedlicher Optimierungsverfahren stellen hybride Optimierungsverfahren oft eine Kombination verschiedener Verfahren dar. Dabei werden in der Regel stochastische und deterministische Verfahren miteinander kombiniert. Es können aber auch rein stochastische oder rein deterministische Kombinationen verwendet werden. Zudem finden statistische Versuchspläne, Metamodelle und Heuristiken Anwendung. Entsprechende Optimierungsalgorithmen sind oft sehr auf den jeweiligen Anwendungsfall spezialisiert.

Häufig bestehen hybride Optimierungsverfahren aus zwei Phasen. In der ersten Phase wird der Designvariablenraum durchsucht. Dies kann durch ein stochastisches Optimierungsverfahren oder durch einen statistischen Versuchsplan erfolgen. Anschließend findet an dem bisher gefundenen Optimum eine detaillierte Optimierung statt, in der Regel durch ein deterministisches Verfahren. Das deterministische Verfahren setzt somit nicht an einer oder mehreren willkürlichen Stellen des Designvariablenraumes an, sondern bereits an einem optimierten Punkt, wodurch die Wahrscheinlichkeit reduziert wird, in einem lokalen Minimum zu verharren. Gleichzeitig wird die hohe Anzahl an Evaluationen reduziert, die ein stochastisches Verfahren für die lokale Suche benötigt [SG15]. Beispielhaft ist die Verwendung des Partikelschwarmverfahrens mit automatischem Wechsel in ein Gradientenverfahren [PP10] sowie der Einsatz eines GA bei gleichzeitiger Erstellung eines Metamodells zur lokalen Optimierung durch ein Gradientenverfahren [NL05] zu nennen. Es können aber auch rein stochastische Verfahren miteinander kombiniert werden [RS12]. Auch bei diesen Verfahren ist der Einsatz verschiedener Parallelisierungsmethoden sinnvoll. So können die unabhängigen Evaluationen der stochastischen Verfahren sowie die Berechnung der Stützstellen der Versuchspläne parallelisiert werden um die Effizienz weiter zu steigern (s. Kapitel 4).

2.7 Explorative Untersuchung des Lösungsraumes

Die explorative Untersuchung des Lösungsraumes ist alternativ bzw. als Ergänzung oder Vorbereitung einer Optimierung zu sehen, insbesondere wenn im Vorfeld einer Optimierungsaufgabe nur wenig Informationen und Wissen über den vorhandenen Lösungsraum und die geltenden Zusammenhänge zur Verfügung stehen. Im Gegensatz zur Optimierung, bei der der Lösungsraum gezielt nach seinem Optimum bzw. nach der Menge Pareto-optimaler Lösungen durchsucht wird, wird hier der gesamte Lösungsraum erkundend (explorativ) untersucht und das Zielfunktionsverhalten analysiert. Dabei wird der Lösungsraum in der Regel durch mathematische Modelle approximiert, welche als Metamodelle, Response-Surface-Modelle (RSM), Surrogate-Modelle, Regressionsmodelle oder als Ersatzmodelle bezeichnet werden.

Nach der Auswahl des zu verwendenden Metamodells wird mit Verfahren der statistischen Versuchsplanung (Design of Experiments, DoE) ein Versuchsplan erstellt, welcher anhand definierter Stützstellen (sample points) den gesamten Designvariablenraum abtastet. Basierend auf den Stützstellen wird das gewählte Metamodell angepasst. Dies kann z. B. durch die Anpassung der Koeffizienten der Approximationsfunktion mithilfe der Fehlerquadratrechnung erfolgen [Sch13].

Durch die Verwendung von Metamodellen kann die Anzahl rechenintensiver Simulationen (z. B. Crash-Simulationen) in Optimierungsstudien reduziert werden, da die Simulationen lediglich als Stützstellen und zur Kontrolle des Modells verwendet werden und die Optimierung auf Basis des Metamodells erfolgt [SPKA01]. Insbesondere bei komplexen multidisziplinären Optimierungen kann somit der numerische Aufwand deutlich reduziert werden. Dabei können die lastfall- und fachdisziplinspezifischen Metamodelle in den jeweiligen Abteilungen erstellt und anschließend in eine umfassende multidisziplinäre Optimierung eingebunden werden [SK00], [RBN15]. Dies stellt die häufigste Anwendung von Metamodellen in der Entwicklung technischer Produkte dar [SPKA01]. Abbildung 2.26 zeigt beispielhaft die Reduzierung der Laufzeit der Optimierung eines Fahrzeugrohbaus

durch die Verwendung von Metamodellen. Es ist zu erkennen, dass die Laufzeit durch die Verwendung von RSM bzw. verschiedener RSM in sequentieller Reihenfolge (Seq. RSM) im Gegensatz zu einer gradientenbasierten Optimierung mit mehreren Startpunkten und einem Genetischen Algorithmus (GA) deutlich reduziert werden kann. Das beste Ergebnis bei der Minimierung der Masse des Rohbaus liefern die sequentiellen RSM und der GA.

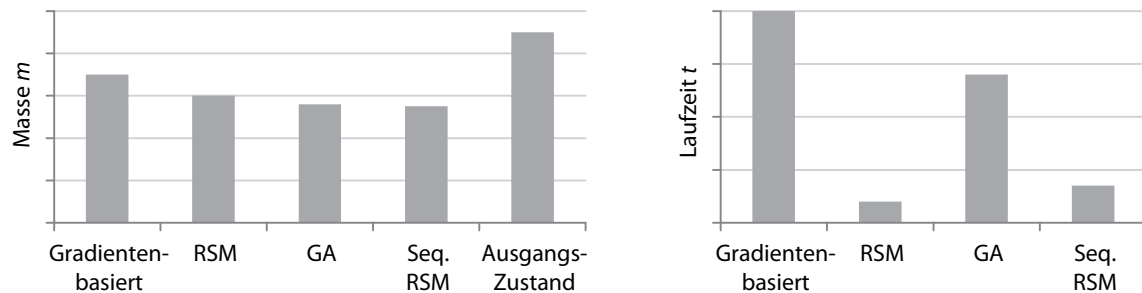


Abbildung 2.26: Reduzierung der Laufzeit einer Optimierung durch den Einsatz von Metamodellen, nach [SHC11]

Weiterhin können durch die mathematische Beschreibung des Lösungsraumes relevante Parameter und deren Einfluss identifiziert werden, was insbesondere bei der Neukonstruktion und Konzeptentwicklung das Systemverständnis für das zu entwickelnde Produkt erhöht, aber auch bei der Auswahl von Designvariablen für eine nachfolgende Optimierung hilfreich ist. Bei einer laufenden Optimierung kann die parallele Erstellung eines Metamodells und dessen Verwendung zudem lange Wartezeiten aufgrund von belegten oder ausgefallenen Ressourcen reduzieren, da in diesen Situationen spontan auf das Metamodell zurückgegriffen wird [RB13].

2.7.1 Design of Experiments

Die statistische Versuchsplanung (Design of Experiments, DoE) beinhaltet die Ermittlung der grundlegenden Zusammenhänge zwischen unabhängigen Variablen (Designvariablen) und abhängigen Variablen (Zielgrößen). Da aufgrund der Vielzahl und der Wertebereiche der Designvariablen nicht alle möglichen Kombinationen durch Versuche bzw. Simulationen überprüft werden können, ist das Ziel der statistischen Versuchsplanung eine möglichst genaue Vorhersage der Zusammenhänge bei einer möglichst niedrigen Anzahl an Versuchen. Versuchspläne sollten immer bezogen auf das jeweilige abzubildende System und nach der Art des gewählten Metamodells ausgewählt werden. Keiner der etablierten Versuchspläne kann generell bevorzugt werden. Die Auswahl des besten Versuchsplanes ist sehr schwierig und benötigt oft mehrere Iterationen [Sch13]. Tabelle 2.7 gibt einen Überblick über die etablierten Versuchspläne.

Tabelle 2.7: Charakteristik und Stützstellen etablierter Versuchspläne, nach [SBH10], [Sch13], [Lee14]

Charakteristik	Versuchsplan	Stützstellen	$n = 2$	$n = 3$	$n = 10$
Faktoriell	Vollfaktoriell	$n_p = l^n$ $l = 2$:	4	8	1024
		$l = 3$:	9	27	59049
	Teilfaktoriell	$n_p = l^{n-r}$	halb belegt bei $l = 2$ und $r = 1$		
	Plackett-Burman	festgelegt	4	4	12
Response-Surface	Box-Behnken	festgelegt	1	13	181
	Central-Composite	$n_p = 2^n + 2n + 1$	9	15	1045
Zufallsbasiert	Monte-Carlo	beliebig	bel.	bel.	bel.
	Latin-Hypercube	beliebig	bel.	bel.	bel.

Statistische Versuchspläne werden gemäß ihrer Charakteristik unterschieden. Faktorielle Versuchspläne basieren auf festgelegten Tabellen, welche die Position und somit die Anzahl der Stützstellen (sample points) n_p definieren. Eine Stützstelle bestimmt hierbei eine Variante der Designvariablenwerte. Response-Surface-Versuchspläne eignen sich in erster Linie für die Erzeugung von Metamodellen durch RSM, da hierbei nicht nur die Ecken des Designvariablenraumes überprüft werden, sondern zusätzliche Stützstellen innerhalb und auch außerhalb des Raumes verwendet werden. Bei zufallsbasierten Versuchsplänen sind die Stützstellen nicht im Vorfeld festgelegt. Die Auswahl der Stützstellen erfolgt zufallsbasiert.

Die Anzahl der Stützstellen n_p hängt vor allem von der Anzahl der Designvariablen n ab. Bei voll- und teilfaktoriellen Versuchsplänen, in denen die Anzahl zusätzlich von der Anzahl der Stufen l abhängt, sowie bei Central-Composite-Versuchsplänen kann die Anzahl der Stützstellen analytisch bestimmt werden. Andernfalls ist die Anzahl durch den Versuchsplan vorgegeben oder kann bei zufallsbasierten Versuchsplänen beliebig gewählt werden.

Da der Versuchsplan ein lineares Gleichungssystem darstellt, in dem jeder Versuch eine Gleichung generiert, ist es möglich, Metamodelle anzupassen, deren Designvariablenanzahl der Zahl der Versuche entspricht. Generell ist jedoch ein Überschuss an Gleichungen anzustreben, da somit eine Kontrolle des erstellten Modells möglich ist [SBH10].

Nachfolgend werden die etablierten Versuchspläne mit ihrer Charakteristik und möglichen Anwendungsfeldern beschrieben:

Voll- und teilfaktorielle Versuchspläne: Vollfaktorielle Versuchspläne stellen die ursprüngliche und einfachste Art der Versuchspläne dar. In einem zweistufigen vollfaktoriellen Versuchsplan ($l = 2$) werden ausschließlich die Ecken des Designvariablenraumes als Stützstellen verwendet. Die Anzahl der Stützstellen hängt also lediglich von der Anzahl der Designvariablen n ab. In einigen Fällen kann auch zusätzlich der Mittelpunkt des Designvariablenraumes als weitere Stützstelle verwendet werden [Cav13]. Zweistufige vollfaktorielle Versuchspläne eignen sich in erster Linie für die Erstellung *linearer* Beschreibungsmodelle, stellen aber auch eine einfache Möglichkeit dar, die Stabilität eines parametrischen Modells im Vorfeld einer Optimierung zu überprüfen.

Ist ein lineares Beschreibungsmodell nicht ausreichend, kann ein dreistufiger Versuchsplan ($l = 3$) zur Erstellung eines *quadratischen* Beschreibungsmodells verwendet werden. Dabei werden zwischen den Eckpunkten weitere Stützstellen platziert, wie in Abbildung 2.27 dargestellt ist. Die Berechnung der Stützstellenanzahl erfolgt gemäß der Gleichung $n_p = l^n$ (s. Tabelle 2.7).

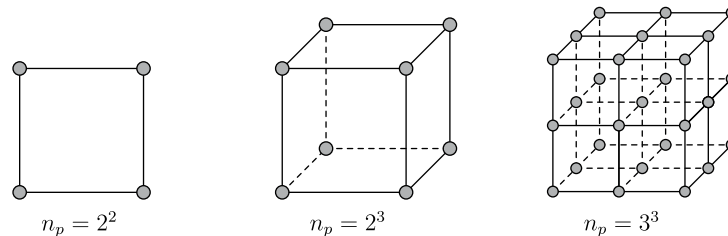


Abbildung 2.27: Vollfaktorielle Versuchspläne, nach [Cav13]

Durch die Erhöhung der Stufenanzahl l können auch kubische oder höhergradige Polynome zur Metamodellgenerierung verwendet werden. Eine Erhöhung der Designvariablenanzahl n oder der Stufen führt zu einer sehr starken Erhöhung der Anzahl an benötigten Stützstellen (s. Tabelle 2.7), sodass die Verwendung vollfaktorieller Versuchspläne nicht mehr effizient ist [Cav13]. Aus diesem Grund werden vollfaktorielle Versuchspläne meist nicht vollständig berechnet, sondern es werden durch eine Reduzierung der Stützstellen teilfaktorielle Versuchspläne erstellt.

Die Grundlage für diese Reduzierung liegt in der Abbildung der Haupteffekte und Wechselwirkungen der Variablen durch das Metamodell. Aus einem vollfaktoriellen Versuchsplan können alle Haupteffekte und Wechselwirkungen der Designvariablen untereinander abgeleitet werden. Ein zweistufiger Versuchsplan mit drei Designvariablen (2^3), welcher acht Stützstellen beinhaltet, berücksichtigt neben den drei Haupteffekten der drei Designvariablen auch drei Zweifachwechselwirkungen der Designvariablen untereinander und eine Dreifachwechselwirkung aller Designvariablen. Diese sieben Effekte werden auch als Freiheitsgrade bezeichnet. Ein zweistufiger vollfaktorieller Versuchsplan besitzt immer $2^n - 1$ Freiheitsgrade [MMAC09].

Durch die Vernachlässigung der Dreifachwechselwirkungen kann dem Versuchsplan bei gleicher Stützstellenanzahl eine neue Variable hinzugefügt werden. Dies führt dazu, dass sich die Haupteffekte und die durch die vierte Designvariable erzeugten Dreifachwechselwirkungen überlagern. Da die Terme höherer Ordnung vernachlässigt werden, können die Haupteffekte sicher bestimmt werden. Lediglich die Zweifachwechselwirkungen lassen sich nicht eindeutig zuordnen, was aufgrund der geringen Anzahl an Stützstellen jedoch einen guten Kompromiss darstellt, da die Haupteffekte der Designvariablen höchste Priorität haben. Auf diese Weise können bereits bei einer hohen Anzahl von Designvariablen Metamodelle durch wenige Stützstellen erzeugt werden.

Die Berechnung der Stützstellenanzahl teilfaktorieller Versuchspläne erfolgt gemäß der Gleichung $n_p = l^{n-r}$ (s. Tabelle 2.7). Der Faktor r repräsentiert dabei die Reduktionsstufe, um die der ursprüngliche vollfaktorielle Versuchsplan reduziert wird. Abbildung 2.28 zeigt die Reduzierung der in Abbildung 2.27 dargestellten dreidimensionalen vollfaktoriellen Versuchspläne bei einer Reduktionsstufe von $r = 1$.

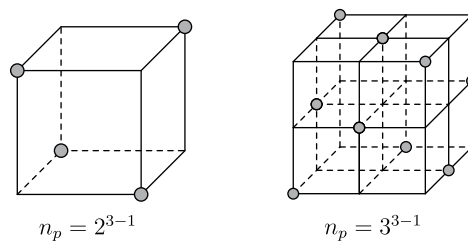


Abbildung 2.28: Teilfaktorielle Versuchspläne, nach [Cav13]

Die Bezeichnung von teilfaktoriellen Versuchsplänen erfolgt durch die Angabe der möglichen Kombinationen l^n und der Reduktionsstufe r . Ein zweistufiger Versuchsplan mit 7 Designvariablen und einer Reduktionsstufe von 4 wird durch den Term 2^{7-4} gekennzeichnet. Dieser oft verwendete Versuchsplan benötigt dabei lediglich acht Stützstellen ($2^{7-4} = 2^3 = 8$). Zweistufige teilfaktorielle Versuchspläne verwenden bei einer Reduktionsstufe von 1 im Gegensatz zu vollfaktoriellen Versuchsplänen immer 50 % der Stützstellen. Bei einem durch den Term 2^{5-1} gekennzeichneten Versuchsplan entspricht dies 16 Stützstellen [SBH10].

Weitere erfolgreich verwendete teilfaktorielle Versuchspläne sowie eine detaillierte Beschreibung der Effekt- und Wechselwirkungsberechnung sind in [MMAC09], [SBH10] und [Cav13] beschrieben.

Plackett-Burman: Die von Plackett und Burman [PB46] entwickelten zweistufigen Versuchspläne sind durch sehr wenige Stützstellen gekennzeichnet, wobei in den Ergebnissen stets die Haupteffekte mit Anteilen der Zweifachwechselwirkungen überlagert sind. Die Vermengungsstruktur von Versuchsplänen wird durch die sog. Auflösung beschrieben. Die beschriebene Vermengung von Haupteffekten und Zweifachwechselwirkungen bei Plackett-Burman-Versuchsplänen entspricht einer Auflösungsstufe von III. Die Wechselwirkungen wirken sich bei diesen Versuchsplänen nicht vollständig auf die Berechnung der Haupteffekte aus. Es werden aber alle Haupteffekte der nicht an der jeweiligen Zweifachwechselwirkung beteiligten Designvariablen verfälscht, was aufgrund der hohen Effizienz toleriert wird [SBH10].

Die Anzahl der Stützstellen ist bei diesen Versuchsplänen immer ein Vielfaches von 4 und deckt einen Bereich von 4–96 ab. Dabei können maximal $n = n_p - 1$ Parameter durch einen Versuchsplan abgedeckt werden [Cav13]. Ein Plackett-Burman-Versuchsplan mit 12 Versuchen (L12) kann also mit maximal 11 Designvariablen verwendet werden. Obwohl dabei weniger als 0,6 % der möglichen Kombinationen ausgewertet werden, können in der Regel die signifikanten Haupteffekte der Designvariablen identifiziert werden. Die Effizienz dieser Versuchspläne steigt mit der Anzahl der Variablen an. So werden in einem L20-Versuchsplan mit 19 Designvariablen lediglich 0,0038 % der möglichen Kombinationen als Stützstellen verwendet [SBH10]. Die Versuchspläne L4, L8, L16 und L32 entsprechen den teilfaktoriellen Versuchsplänen 2^{3-1} , 2^{7-4} , 2^{15-11} und 2^{31-26} [Cav13].

Plackett-Burman-Versuchspläne eignen sich in erster Linie zur Ermittlung der Haupteffekte der Designvariablen und können somit zur Identifizierung geeigneter Designvariablen im Vorfeld einer Optimierung eingesetzt werden.

Box-Behnken: Versuchspläne nach Box und Behnken sind unvollständige dreistufige vollfaktorielle Versuchspläne. Die Stützstellen werden dabei auf den Kanten des Designvariablenraumes positioniert. Die Ecken werden nicht betrachtet [BB60]. Hinzu kommt eine zentrale Stützstelle im Mittelpunkt. Bei statistischen Versuchen, in denen eine Streuung der Ergebnisgrößen zu erwarten ist, werden mehrere Versuchsdurchläufe am Mittelpunkt durchgeführt, um die Streuung abschätzen zu können. Bei deterministischen Versuchen, z. B. durch den Einsatz von CAE, genügt ein Durchlauf am Mittelpunkt. Drei Designvariablen resultieren somit in 13 Stützstellen, wie in Abbildung 2.29 dargestellt ist [SBH10].

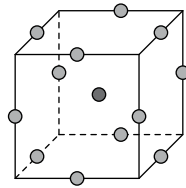


Abbildung 2.29: Box-Behnken-Versuchsplan, nach [SBH10]

Ein Box-Behnken-Versuchsplan ist vor allem sinnvoll, wenn ein quadratisches Beschreibungsmodell angestrebt wird und die Ecken des Designvariablenraumes kritische Bereiche darstellen. Da die Ecken in dem Versuchsplan nicht berücksichtigt werden, ist das Beschreibungsmodell nicht in den Eckbereichen gültig. Die Ecken liegen außerhalb des untersuchten Bereichs und Extrapolationen sind in der Regel nicht zulässig, da sich außerhalb des untersuchten Bereichs das Systemverhalten sprunghaft ändern kann. Ist bei bekannten nichtlinearen Zusammenhängen das Optimum in der Mitte des Designvariablenraumes zu erwarten, können die Nachteile dieses Versuchsplanes vernachlässigt werden. Der Box-Behnken-Versuchsplan wird aufgrund seiner Effizienz uneingeschränkt für drei bis fünf Designvariablen empfohlen. Bei einer hohen Anzahl an Designvariablen sind die Nachteile zu berücksichtigen [SBH10].

Central-Composite: Central-Composite-Versuchspläne basieren auf einem zweistufigen Versuchsplan, dem ein Mittelpunkt und eine n -dimensionale sternförmige Struktur hinzugefügt werden. Durch diese Trennung besteht die Möglichkeit, zunächst den zweistufigen Versuchsplan zu analysieren und bei Bedarf die weiteren Versuchsläufe durchzuführen. Central-Composite-Versuchspläne werden gemäß des Verhältnisses zwischen dem zweistufigen Versuchsplan und der sternförmigen Struktur in drei Arten unterschieden: Central-Composite-Circumscribed (CCC), Central-Composite-Faced (CCF) und Central-Composite-Inscribed (CCI). Diese sind für zwei- und dreidimensionale Probleme in Abbildung 2.30 dargestellt und werden im Folgenden erläutert.

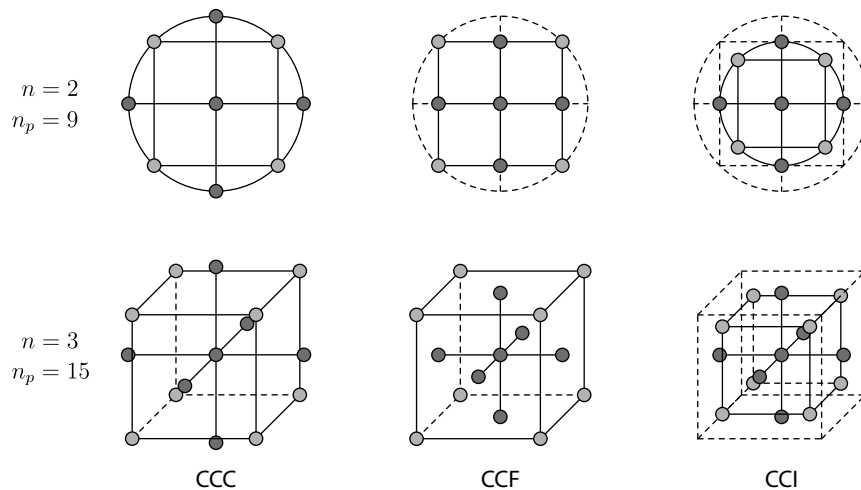


Abbildung 2.30: Central-Composite-Versuchsplan, nach [SBH10], [Cav13]

Die sternförmige Struktur entsteht durch die Variation der Designvariablen ausgehend vom Mittelpunkt. Dabei liegen diese Stützstellen außerhalb der würfelförmigen Struktur des zweistufigen Versuchsplans und jede Variable wird auf fünf Stufen getestet. Dieser kreis- bzw. kugelförmig begrenzte Versuchsplan wird als *Central-Composite-Circumscribed* (CCC) bezeichnet.

Da der über den Würfel hinausragende Stern aufgrund der Variablen Grenzen oft nicht realisierbar ist, werden die sternförmigen Stützstellen auf die Flächen des Würfels projiziert (*Central-Composite-Faced*, CCF). Obwohl die Anzahl der Stützstellen gleich bleibt, wird in diesem Fall jede Designvariable nur noch auf drei Stufen analysiert, was zu einer Verschlechterung der Aussagekraft des Modells führt, da die quadratischen Effekte untereinander korrelieren [SBH10].

Abhilfe schafft die Begrenzung des Versuchsplans durch den Innenkreis bzw. die Kugel innerhalb des Würfels (*Central-Composite-Inscribed*, CCI). Dadurch entsteht ein verkleinerter Würfel und die Anzahl der getesteten Stufen erhöht sich wieder auf 5.

Central-Composite-Versuchspläne werden vor allem bei Problemen mit wenig Designvariablen und zur Ermittlung quadratischer Beschreibungsmodelle verwendet [Sch13]. Die Anzahl der benötigten Stützstellen steigt nur moderat mit der Anzahl der verwendeten Designvariablen an und die Eigenschaften der durch den Versuchsplan erstellten Felder sind generell sehr gut, was zu einer häufigen Anwendung dieser Versuchspläne führt [AW05].

Monte-Carlo: Monte-Carlo-Versuchspläne stellen die einfachste Form der zufallsbasierten Versuchsplanung dar, welche auch als *Space Filling* bezeichnet wird und darauf abzielt, den Designvariablenraum möglichst raumfüllend zu untersuchen [Sch13].

Die Designvariablenwerte werden unabhängig voneinander durch einen Zufallsgenerator ermittelt. Dadurch entstehen bei ausreichend großen Feldern nur schwache Korrelationen der Variablen und es müssen meist sehr viele Stützstellen ermittelt werden, wodurch das Verfahren weniger effizient ist als spezielle problemspezifische Versuchspläne [SBH10]. Abbildung 2.31 zeigt einen Monte-Carlo-Versuchsplan mit 100 Stützstellen.

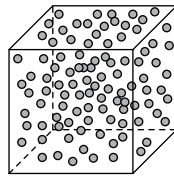


Abbildung 2.31: Monte-Carlo-Versuchsplan mit 100 Stützstellen

Monte-Carlo-Versuchspläne werden zur Erzeugung quadratischer Beschreibungsmodelle verwendet, insbesondere wenn die Anzahl der Versuchsläufe vernachlässigt werden kann, da z. B. sehr schnelle CAE-Modelle mit automatisierter Ablaufsteuerung verwendet werden [SBH10]. Aufgrund der zufälligen Streuung der Designvariablenwerte können die Versuchspläne auch zur Untersuchung der Robustheit von Modellen verwendet werden [Cav13].

Latin-Hypercube: Latin-Hypercube-Versuchspläne reduzieren die Anzahl der Stützstellen im Vergleich zu Monte-Carlo-Versuchsplänen um 50% ohne Einschränkungen der Vorteile und der Ergebnisqualität, da die zufällige Verteilung der Stützstellen im Designvariablenraum gesteuert und somit eine gleichmäßige Verteilung gewährleistet wird. Wie in Abbildung 2.32 dargestellt, wird der gesamte Designvariablenraum in Zeilen und Spalten eines gleichmäßigen Gitters unterteilt, in dem jede Spalte aus einer zufälligen Permutation besteht. Somit wird sichergestellt, dass in jeder Spalte und jeder Zeile jeweils nur genau eine Stützstelle entsteht [SBH10].

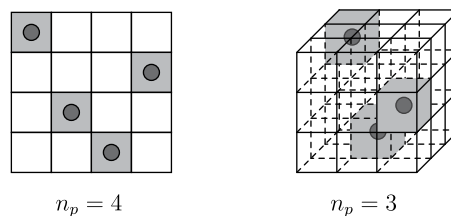


Abbildung 2.32: Latin-Hypercube-Versuchsplan, nach [Cav13]

Da Latin-Hypercube-Versuchspläne sehr wenige Stützstellen verwenden und gleichzeitig viele Stufen des Designvariablenraumes abbilden, sind sie sehr effizient und werden sehr oft zur Erstellung nichtlinearer Metamodelle verwendet, insbesondere wenn numerisch aufwendige Simulationen verwendet werden, z. B. Crash-Simulationen. Die Anzahl der Stützstellen sollte mindestens dem Dreifachen der Designvariablenanzahl entsprechen ($n_p \geq 3n$) [YWT⁺05], [SYZ12], [RBN15].

2.7.2 Generierung von Metamodellen

Basierend auf den durch die statistische Versuchsplanung erzeugten Stützstellen kann der Lösungsraum durch Ersatzmodelle approximiert werden. Diese werden als Metamodelle bezeichnet. Aufgrund der steigenden Komplexität der verwendeten numerischen Simulationen ist die Bedeutung und die Anwendung von Metamodellen in den letzten 15 Jahren stetig gestiegen [Toa15]. Ausgehend von einer begrenzten Anzahl an Designvariablen und

Lastfällen in Verbindung mit Polynomen niedrigen Grades und gradientenbasierten Optimierungsverfahren [CSDV02] haben sich die Verfahren zur Generierung von Metamodellen stets weiterentwickelt. So werden bei komplexeren Anwendungsfällen weitaus mehr Designvariablen und Lastfälle berücksichtigt und komplexe Metamodelle in Kombination mit stochastischen Optimierungsverfahren verwendet [SHC11],[RBN15].

Zur lokalen Approximation bestimmter Bereiche des Lösungsraumes sind in der Regel bereits lineare Modelle ausreichend [MMAC09]. Soll der gesamte Lösungsraum durch ein globales Modell approximiert werden, werden komplexere Metamodelle wie Kriging, Radiale Basisfunktionen oder Neuronale Netzwerke verwendet, welche im Folgenden beschrieben werden. Für weiterführende Informationen zu den mathematischen Grundlagen und der Generierung von Metamodellen wird auf [SPKA01], [SLC01] [MMAC09], [FK09], [SBH10], [Cav13], [RBN15] verwiesen.

Bevor ein Überblick über die am häufigsten verwendeten Metamodelle gegeben wird, werden nachfolgend kurz die mathematischen Grundlagen der Metamodellgenerierung erläutert:

Die Zielfunktion $f(\mathbf{x})$ ist in der Regel eine im Vorfeld der Optimierung nicht bekannte Funktion der Designvariablen \mathbf{x} (s. Abschnitt 2.2). Diese Zielfunktion wird nun durch ein Metamodell $\hat{f}(\mathbf{x})$ approximiert:

$$y = f(\mathbf{x}) = \hat{f}(\mathbf{x}) + \epsilon(\mathbf{x}) \quad \Rightarrow \quad \hat{y} = \hat{f}(\mathbf{x}) \quad (2.14)$$

Dabei stellt $\epsilon(\mathbf{x})$ den Fehler in der Approximation dar. Dieser Fehler repräsentiert den Teil des zu beschreibenden Systems, der nicht durch das Metamodell beschrieben werden kann [SBH10], [Cav13]. Die Ergebnisse eines Versuchsplans mit n_p Stützstellen sind n_p Wertepaare (\mathbf{x}_i, y_i) . Jedes Wertepaar ordnet dabei einem Punkt \mathbf{x}_i des Designvariablenraumes einen Wert des Lösungsraumes zu (s. Abschnitt 2.4). Werden wie bei einer multikriteriellen Optimierung (s. Abschnitt 2.8.1) mehrere Zielgrößen ausgewertet, so wird für jede Zielgröße eine eigene mathematische Beschreibung verwendet. Die grafische Darstellung der Metamodelle erfolgt in der Regel über Antwortflächen (RSM) oder Konturdiagramme.

Lineare Regression: Die lineare Regression stellt die einfachste Form der Metamodellgenerierung dar. Aufgrund der einfachen Anwendung und der umfangreichen Literatur wird die lineare Regression bei vielen Analysen und Studien eingesetzt. Insbesondere wenn die grundlegenden Zusammenhänge zwischen Designvariablen und Zielgrößen bereits bekannt sind, können mit geringem Aufwand ausreichend genaue Metamodelle erzeugt werden.

Das allgemeine lineare Regressionsmodell wird durch die folgende Gleichung repräsentiert:

$$\hat{y} = b_0 + b_1 \cdot x_1 + b_2 \cdot x_2 + \dots + b_n \cdot x_n \quad (2.15)$$

Dabei werden die Faktoren b_0, \dots, b_n so angepasst, dass das Modell mit den Stützstellen aus der statistischen Versuchsplanung übereinstimmt. Vor allem bei komplexeren Systemen sollte nach der Erstellung die Gültigkeit des Modells überprüft werden. Dazu werden die Abweichungen zwischen den Stützstellen und den Vorhersagen des linearen Regressionsmodells bestimmt. Die auch als Residuen bezeichneten Abweichungen müssen ausreichend geringe Absolutwerte aufweisen und gleichmäßig über alle Stützstellen variieren [SBH10].

Polynome: Polynome stellen die am häufigsten verwendete Form zur Generierung von Metamodellen in der Entwicklung technischer Produkte dar [FK09]. Hierbei werden in der Regel Polynome folgender Form verwendet:

$$\hat{y} = b_0 + b_1 \cdot x_1 + b_2 \cdot x_1^2 + b_3 \cdot x_2 + b_4 \cdot x_2^2 + \dots + b_k \cdot x_1 \cdot x_2 + \dots \quad (2.16)$$

Die Bestimmung der Faktoren b_0, \dots, b_n erfolgt durch die Substitution der nichtlinearen Terme mit Ersatzvariablen, wodurch ein lineares Gleichungssystem entsteht und die Faktoren mit der *Methode der kleinsten Fehlerquadrate* berechnet werden können.

Mit der Erhöhung der Anzahl der Designvariablen oder des Polynomgrades steigt die Anzahl der Terme stark an, was zu einem erhöhten Rechenaufwand für die Faktorbestimmung führt und eine große Anzahl von Stützstellen erfordert. Polynome werden daher meist auf quadratische Terme beschränkt. In modernen Softwaretools stehen aber auch Polynome höherer Ordnungen zur Verfügung [SBH10]. Die Verwendung von Polynomen 2. Grades stellt in Kombination mit Central-Composite-Versuchsplänen und der Methode der kleinsten Fehlerquadrate die häufigste Anwendungsform dar [SPKA01].

Abbildung 2.33 zeigt beispielhaft die Approximation der folgenden Sinusfunktion:

$$y = \sin(\pi \cdot x_1) \cdot x_2 \quad (2.17)$$

Es wird ein Polynom 2. Grades mit folgender Gleichung verwendet:

$$\hat{y}_{Pol} = -0,4807 + 2,3 \cdot x_1 + 0,8279 \cdot x_2 - 2,0420 \cdot x_1^2 + 0,4415 \cdot x_2^2 - 0,8785 \cdot x_1 \cdot x_2 \quad (2.18)$$

Das verwendete Polynom bildet zwar den grundsätzlichen Zusammenhang zwischen den Designvariablen x_1 , x_2 und der Zielgröße y ab, zeigt aber an den Randgebieten deutliche Abweichungen zum Ursprungsmodell der Sinusfunktion. Die zufällig bestimmten Stützstellen sind grau dargestellt.

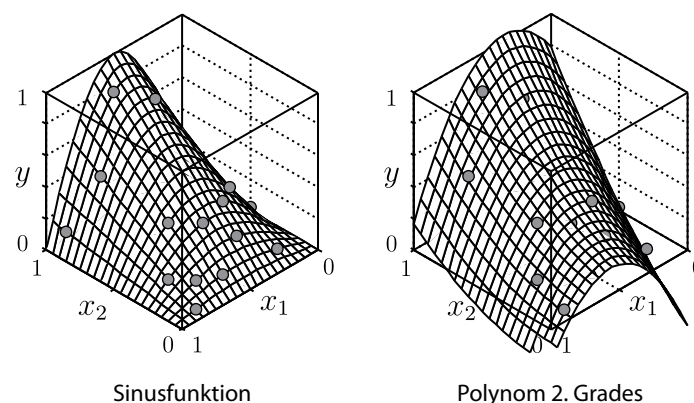


Abbildung 2.33: Approximation einer Sinusfunktion durch ein Polynom 2. Grades, nach [SBH10]

Ist ein quadratisches Beschreibungsmodell nicht ausreichend, kann der Polynomgrad erhöht und z. B. ein kubisches Beschreibungsmodell verwendet werden. In diesem Fall muss der Versuchsplan höhere Anforderungen erfüllen, sodass Versuchspläne mit vielen Zwischenstufen empfohlen werden, z. B. Monte-Carlo oder Latin-Hypercube [SBH10].

Kriging: Beim nach dessen Erfinder Danie G. Krige [Kri51] benannten Kriging-Verfahren erfolgt die Approximation des Metamodells durch Interpolation der Stützstellen in der lokalen Umgebung eines gesuchten Bereichs. Den Stützstellen, welche sich näher an einer gesuchten Stelle befinden, wird eine größere Bedeutung entgegengebracht als entfernteren Stützstellen [Kri51], [SBH10].

Ausgehend von einer Basisfunktion, welche meist den konstanten globalen Mittelwert μ aller Stützstellen abbildet, wird beim *einfachen Kriging* (simple Kriging) der Lösungsraum durch folgende Funktion approximiert [SBH10], [Sch13]:

$$\hat{y} = \mu + \sum_{i=1}^{n_p} (y_i - \mu) \cdot e^{-\theta_i \cdot |x - x_i|^{p_k}} \quad \text{mit} \quad 0 < p_k \leq 2 \quad (2.19)$$

Ist der Mittelwert μ nicht bekannt, wird er durch den Mittelwert aller Stützstellen berechnet. Der Parameter p_k bestimmt die Glättung der Approximationsfunktion. Je größer dieser Wert ist, desto glatter verläuft die Funktion. Bei einem oft verwendeten Wert von $p_k = 2$ werden um jede Stützstelle Gauß-Verteilungen als Basisfunktionen verwendet und die glatteste Approximationsfunktion gebildet [Har14]. Weiterhin wird die Funktion durch den Gewichtungsfaktor θ beeinflusst, welcher meist für alle Stützstellen als konstant angenommen wird. Abbildung 2.34 zeigt die Approximation der bisher verwendeten Sinusfunktion durch einfache Kriging-Modelle verschiedener Gewichtungsfaktoren. Der Gewichtungsfaktor θ variiert dabei zwischen den Werten $\theta = 0,001$ und $\theta = 100$ bei einer Glättung von $p_k = 2$. Es ist zu erkennen, dass durch den Gewichtungsfaktor der Einfluss der Stützstellen verändert wird. Große Werte führen zu einer reinen lokalen Approximation in der Nähe einer Stützstelle. Bei kleinen Werten nähert sich die Funktion dem globalen Mittelwert der Stützstellen an und verläuft harmonischer [SBH10], [Sch13].

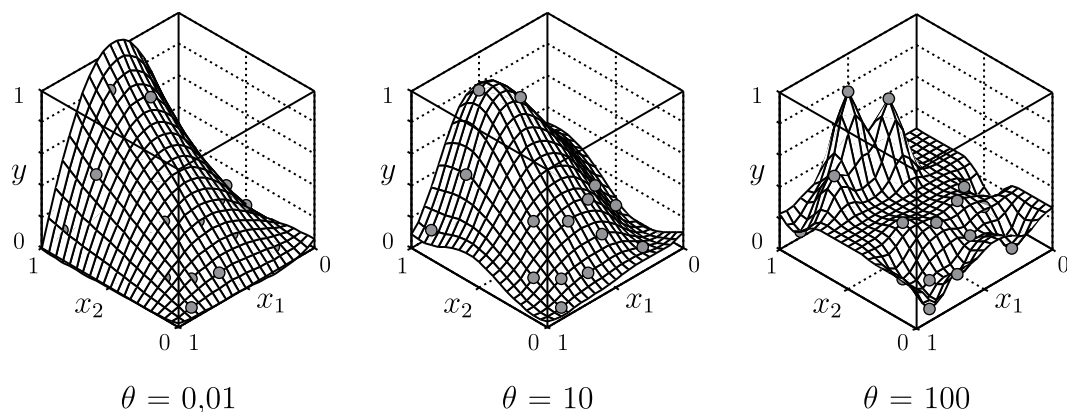


Abbildung 2.34: Approximation einer Sinusfunktion durch Kriging-Modelle unterschiedlicher Gewichtung, nach [SBH10]

Wie in Abbildung 2.34 zu erkennen ist, liefert bereits das einfache Kriging-Modell unter Verwendung eines geringen Gewichtungsfaktors eine sehr gute Approximation der Sinusfunktion und stellt eine gute Alternative zur Approximation durch Polynome dar. In dem dargestellten Beispiel ist die Approximation besser als das zuvor abgebildete Polynom. Insbesondere an den Rändern des approximierten Lösungsraumes besitzen Polynome die Tendenz des Aufschwings [Sch13].

Das anfänglich zur Vorhersage deterministischer Computerexperimente entwickelte einfache Kriging [SWMW89] stellt das ursprüngliche und einfachste Kriging-Verfahren dar und weist ein breites Anwendungsspektrum in der Produktentwicklung und der Bearbeitung von Optimierungsaufgaben auf [Toa15]. Weiterhin bildet das einfache Kriging die Basis einer Vielzahl von Weiterentwicklungen des Kriging-Verfahrens.

Die am häufigsten angewendete Weiterentwicklung ist das *normale Kriging* (ordinary Kriging), welches im Gegensatz zum einfachen Kriging von einem beliebigen und konstanten, aber unbekanntem Mittelwert μ ausgeht. Die benötigten Gewichtungsfaktoren werden unter der Bedingung bestimmt, dass der Approximationsfehler des Metamodells minimiert werden soll und der Erwartungswert des Fehlers einen Wert von 0 annimmt sowie die Summe der Gewichtungsfaktoren gleich 1 ist [SBH10].

Weitere Kriging-Verfahren sind das universelle Kriging, das Gauß'sche Kriging, das nicht-stationäre Kriging, das multi-fidelity Kriging oder das gradient-enhanced Kriging. Für die mathematische Formulierung und die Anwendung dieser Verfahren wird auf [SBH10], [Toa15] verwiesen.

Sind die Stützstellen der zu approximierenden Funktion deterministischer Natur und sollen durch das Metamodell hochgradig nichtlineare Effekte abgebildet werden, ist das Kriging-Verfahren anderen Verfahren vorzuziehen, wobei jedoch nicht mehr als 50 Designvariablen verwendet werden sollten [SPKA01].

Radiale Basisfunktion: Radiale Basisfunktionen beschreiben Funktionen zur exakten Interpolation von Daten in einem mehrdimensionalen Designvariablenraum [Pow87]. Dabei wird eine Linearkombination von Basisfunktionen verwendet, welche über den Abstand eines zu analysierenden Punktes \mathbf{x} zu den gegebenen Stützstellen definiert sind:

$$\hat{y} = \mu + \sum_{i=1}^{n_p} b_i \cdot g_{0i}(\|\mathbf{x} - \mathbf{x}_i\|) \quad (2.20)$$

Der Faktor μ repräsentiert dabei, ähnlich wie beim Kriging, den konstanten globalen Mittelwert aller Stützstellen. Die Faktoren b_i werden durch die Methode der kleinsten Fehlerquadrate ermittelt. Zur Approximation können die Basisfunktionen $g_{0i}(\|\mathbf{x} - \mathbf{x}_i\|)$ verschiedene Grundformen annehmen:

$$\begin{aligned} \text{Linear: } & g_{0i}(\|\mathbf{x} - \mathbf{x}_i\|) = \|\mathbf{x} - \mathbf{x}_i\| \\ \text{Polynom: } & g_{0i}(\|\mathbf{x} - \mathbf{x}_i\|) = \|\mathbf{x} - \mathbf{x}_i\|^k \quad \text{mit } k = 1, 3, 5 \dots \\ \text{Polynom: } & g_{0i}(\|\mathbf{x} - \mathbf{x}_i\|) = \|\mathbf{x} - \mathbf{x}_i\|^k \cdot \log(\|\mathbf{x} - \mathbf{x}_i\|) \quad \text{mit } k = 2, 4, 6 \dots \\ \text{Gauß: } & g_{0i}(\|\mathbf{x} - \mathbf{x}_i\|) = e^{-\theta_i \cdot \|\mathbf{x} - \mathbf{x}_i\|^2} \end{aligned} \quad (2.21)$$

Die Approximation durch Basisfunktionen unter Verwendung von Polynomen unterschiedlichen Grades k ist in Abbildung 2.35 dargestellt. Hierbei weisen alle drei Modelle gute Approximationseigenschaften im Bereich zwischen den Stützstellen auf. In der Extrapolation zu den Randgebieten werden deutliche Abweichungen zum ursprünglichen Verlauf der Sinusfunktion sichtbar. Extrapolation sollten daher im Idealfall vermieden werden. Die Fehler in der Extrapolation der Stützstellen hängen sehr stark vom gewählten Metamodell und den zu approximierenden Zusammenhängen ab. Die Extrapolation sollte jedoch generell vermieden werden [SBH10].

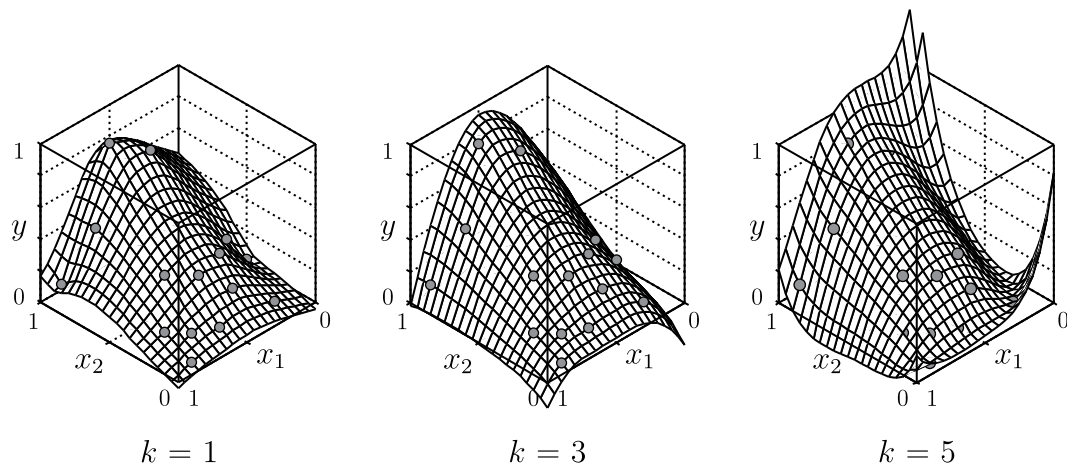


Abbildung 2.35: Approximation einer Sinusfunktion durch unterschiedliche Basisfunktionen, nach [SBH10]

Neuronale Netzwerke: In Neuronalen Netzwerken wird das biologische Nervensystem zur Generierung von Metamodellen simuliert. Dabei können künstliche Neuronen durch einen internen Informationsaustausch untereinander verschiedene Aufgaben übernehmen, z. B. die Approximation eines Funktionswertes y an einem unbekanntem Punkt \mathbf{x} .

Ein Netzwerktyp, welcher häufig zur Generierung von Metamodellen eingesetzt wird, ist das *Feedforward-Netzwerk*, bei dem die Neuronen in unterschiedliche Ebenen aufgeteilt werden. Die erste Ebene bildet die Eingangsebene und enthält die verwendeten Designvariablen \mathbf{x}_i als Eingangssignal. Dieser Ebene gegenüber befindet sich die Ausgangsebene, welche die gesuchte Approximation \hat{y} enthält. Obwohl auch gleichzeitig mehrere Ausgangsgrößen approximiert werden können, ist es in den meisten Fällen sinnvoll, für jede Ausgangsgröße ein separates Neuronales Netzwerk zu verwenden, da das Netzwerk nur so speziell an den zu untersuchenden Zusammenhang angepasst werden kann. Zwischen der Ein- und der Ausgangsebene befinden sich eine oder mehrere versteckte Ebenen. In den meisten Anwendungsfällen sind jedoch nicht mehr als zwei versteckte Ebenen erforderlich [SBH10].

Zur Generierung des Metamodells wird das Neuronale Netzwerk durch die vorhandenen Daten aus den Stützstellen trainiert. Analog zu den biologischen Vorgängen im Gehirn erfolgt die Anpassung der Verbindungen bzw. des Informationsaustauschs zwischen den einzelnen Neuronen der jeweiligen Ebenen. Beim Trainieren des Netzwerkes werden die Gewichtungsfaktoren b_i der Approximation so variiert, dass der quadratische Approximationsfehler minimiert wird. Dies erfolgt meist mittels eines gradientenbasierten Optimierungsverfahrens, z. B. Konjugierte Gradienten (s. Abschnitt 2.6). Da meist viele lokale Minima und Bereiche mit sehr flachen Gradienten existieren, kann die Minimierung des Fehlers sehr komplex und zeitaufwendig sein. Die Minimierung des Approximationsfehlers wird solange fortgesetzt, bis eines der definierten Abbruchkriterien erreicht ist, z. B. ein bestimmter Fehlerwert unterschritten ist, der Approximationsfehler wieder deutlich ansteigt oder eine festgelegte Anzahl an Iterationen erreicht ist [SBH10].

Die Schwierigkeit in der Verwendung von Neuronalen Netzwerken liegt in der richtigen Bestimmung der Anzahl der versteckten Ebenen und Neuronen sowie der Anzahl an notwendigen Iterationen im Trainingsprozess. Eine versteckte Ebene ist in nahezu allen Anwendungsfällen ausreichend, selbst zur Abbildung komplexer Zusammenhänge. Die

Genauigkeit der Approximation wird zwar durch eine zweite versteckte Ebene immer verbessert. Dabei nimmt die Geschwindigkeit des Trainings jedoch deutlich ab. Nur bei sehr komplexen oder nicht kontinuierlichen Zusammenhängen ist eine zweite versteckte Ebene notwendig, z. B. bei einer Sägezahn-Funktion [SBH10].

Abbildung 2.36 zeigt die Approximation der bisher verwendeten Sinusfunktion (s. Abbildung 2.33) durch verschiedene Neuronale Netzwerke mit einer versteckten Ebene in Abhängigkeit von der Neuronenanzahl in der versteckten Ebene (n_{vE1}) und verschiedenen Startwerten für die zu ermittelnden Gewichtungsfaktoren (a und b).

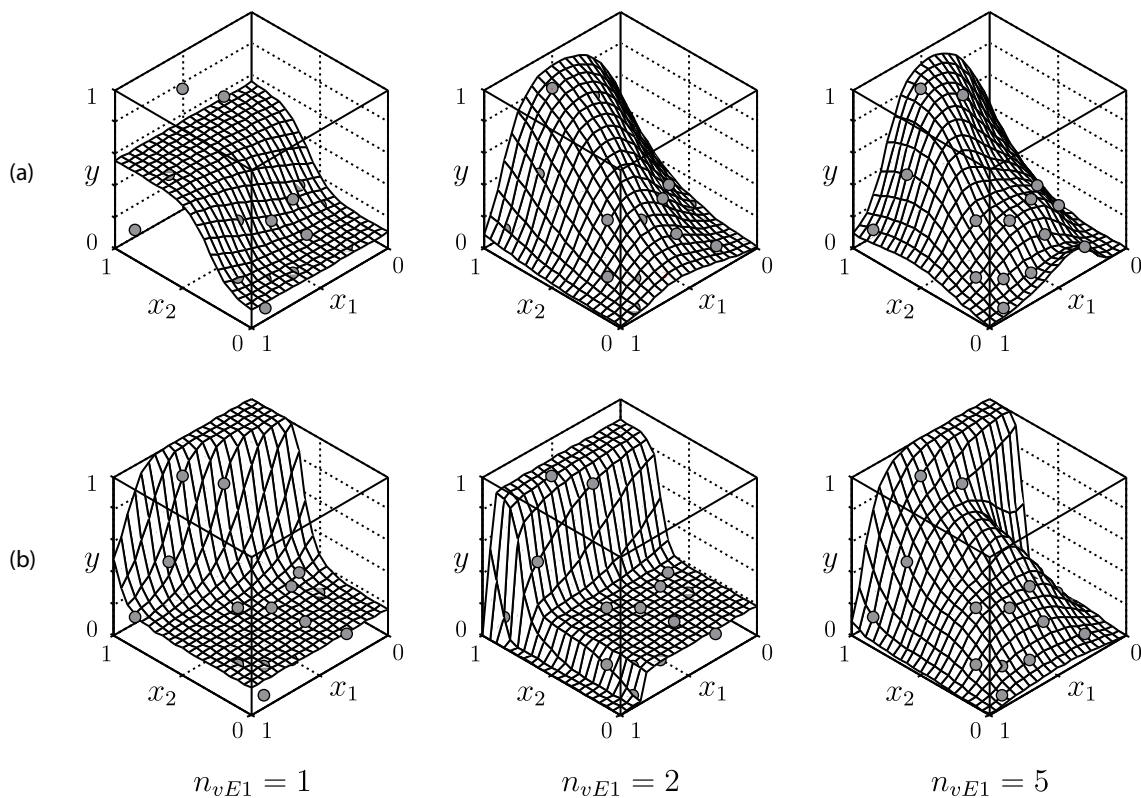


Abbildung 2.36: Approximation einer Sinusfunktion durch Neuronale Netzwerke unterschiedlicher Startgewichte und Neuronenanzahl, nach [SBH10]

Bereits bei zwei Neuronen in der versteckten Ebene wird der gesuchte Zusammenhang ausreichend genau approximiert. Ein Neuron dient lediglich der Abbildung des globalen Trends der Sinusfunktion. Fünf Neuronen steigern die Qualität der Approximation nur geringfügig, erhöhen aber zudem die Gefahr des Overfittings, da mehr Gewichtungsfaktoren als Stützstellen für das Training existieren. Weiterhin ist die Anfälligkeit des Trainings für lokale Minima zu erkennen. Bei der Verwendung von anderen Startwerten für die Gewichtungsfaktoren (b) wird ein deutlich anderer Zusammenhang zwischen den Ein- und den Ausgangsvariablen generiert [SBH10].

Durch die Verwendung von Neuronalen Netzwerken können komplexe Zusammenhänge generell gut abgebildet werden. Bereits einfache Feedforward-Netzwerke mit einer oder zwei versteckten Ebenen sind für fast alle Anwendungsfälle ausreichend und bieten eine gegen Störgrößen und einzelne Fehler in den Stützstellen robuste Methode zur Erzeugung von Metamodellen [SBH10]. Insbesondere wenn sehr viele Designvariablen in determinis-

tischen Analysen verwendet werden, stellen Neuronale Netzwerke eine Alternative zu den bisher vorgestellten Verfahren zur Generierung von Metamodellen dar [SPKA01].

Die Festlegung der optimalen Netzwerkeinstellungen ist jedoch oft schwierig und mit mehreren Anpassungen und Iterationen verbunden. Zudem muss das ermittelte Metamodell in jedem Fall auf seine Allgemeingültigkeit überprüft werden [SBH10]. Das Training eines Neuronalen Netzwerkes verlangt für eine gute Approximation oft sehr viele Stützstellen, welche meist mit einem Monte-Carlo-Versuchsplan erzeugt werden. Im Gegensatz zur Erkennung von Mustern in der Wirtschaftswissenschaft oder der Psychologie spielen Neuronale Netzwerke in der Entwicklung und Optimierung technischer Produkte eine eher untergeordnete Rolle, da sie die an sie gestellten Erwartungen oft nicht erfüllen und die Zusammenhänge mechanischer Probleme nicht korrekt abgebildet werden [Sch13] [Cav13].

2.8 Optimierungsstrategien

Die Optimierungsstrategie beschreibt den Ansatz und das Vorgehen zum Lösen einer Optimierungsaufgabe. Dies beinhaltet die Definition der Designvariablen, der Zielfunktion und der Restriktionen. Je nach zu lösender Aufgabe werden die Optimierungsmethode, das Optimierungsverfahren und der passende Optimierungsalgorithmus ausgewählt. Eine Optimierungsstrategie kann zudem die Kombination verschiedener Optimierungsmethoden und -verfahren beinhalten. Die Festlegung einer geeigneten Optimierungsstrategie ist entscheidend für die Durchführung einer effektiven und effizienten Optimierung.

Ausgehend von der Identifizierung und Kenntnis des Optimierungsproblems sollten bei der Festlegung der Optimierungsstrategie folgende Punkte berücksichtigt werden:

- **Art des Evaluationsmodells:** Das Evaluationsmodell repräsentiert die zur Lösung des Optimierungsproblems notwendige mathematische Beschreibung. Diese kann in *analytischer* oder *numerischer* Form vorliegen. Zudem richtet sich nach der Art des Evaluationsmodells auch die Art der zu erwartenden Lösung. Basiert die Evaluation ausschließlich auf einem Finite-Elemente-Modell, liegt auch die Lösung des Optimierungsproblems in dieser Form vor. Wird zur Evaluation ein parametrisches CAD-Modell verwendet, wird auch die Lösung durch ein CAD-Modell abgebildet.

Weiterhin wird das Evaluationsmodell durch die Definition der Designvariablen und die gewählte Optimierungsmethode beeinflusst. So basiert z. B. die Topologieoptimierung (s. Abschnitt 2.5.2) in der Regel rein auf dem FE-Modell, was die implizite Definition der Designvariablen und die Evaluation auf Basis des FE-Modells bedingt.

Der numerische Aufwand und die Komplexität des Evaluationsmodells können zudem die Auswahl des Optimierungsverfahrens beeinflussen. Führt die Verwendung eines sehr komplexen Evaluationsmodells zu sehr langen Laufzeiten und kann die Gesamtlaufzeit der Optimierung aufgrund begrenzter Ressourcen nicht durch Parallelisierung verkürzt werden, ist die Verwendung eines deterministischen Optimierungsverfahrens effizienter, da im Gegensatz zu stochastischen Optimierungsverfahren weniger Evaluationen benötigt werden [SG15]. Da in diesem Fall auch das Erreichen eines lokalen Optimums eine Verbesserung des Ausgangszustandes darstellt, ist dies ebenfalls als Erfolg zusehen, auch wenn mit dem globalen Optimum eventuell noch eine bessere Lösung existiert [Har14]. Alternativ hierzu ist die Verwendung eines Metamodells zu nennen, welches

auf den Stützstellen eines DoE-Versuchsplans basiert (s. Abschnitt 2.7). Da durch die Metamodellgenerierung das Evaluationsmodell in analytischer Form vorliegt und die Evaluationszeiten somit sehr gering sind, kann ein Optimierungsverfahren eingesetzt werden, welches eine große Anzahl an Evaluationen erfordert.

- **Art der Designvariablen:** Die Designvariablen können in ihrer Art *kontinuierlich*, *diskret (ganzzahlig)* oder *kombinatorisch* sein. Die Art der Designvariablen bedingt in der Regel die Wahl des Optimierungsverfahrens. So eignen sich deterministische Verfahren zur Verwendung von kontinuierlichen Designvariablen. Stochastische Verfahren haben Vorteile in der Verwendung diskreter und kombinatorischer Designvariablen (s. Abschnitt 2.6).
- **Art der Zielfunktion:** Die Zielfunktion kann *linear*, *quadratisch*, *konvex* oder *nicht konvex* sein. Soll sichergestellt werden, dass unter Verwendung eines deterministischen Optimierungsverfahrens das globale Optimum gefunden wird, muss die Zielfunktion konvex sein (s. Abschnitt 2.2).

Da bei der Bearbeitung einer Optimierungsaufgabe die Zielfunktion in der Regel nur an einzelnen Stellen bekannt ist und keine Informationen über den gesamten Verlauf der Funktion vorliegen, kann keine Aussage über die Konvexität getroffen und das globale Optimum nicht sicher bestimmt werden. In diesem Fall wird durch die Verwendung stochastischer oder hybrider Optimierungsverfahren die Wahrscheinlichkeit erhöht, das globale Optimum zu finden.

- **Anzahl der Zielkriterien:** Gemäß der Anzahl der Zielkriterien einer Optimierung wird diese als *monokriteriell* oder *multikriteriell* bezeichnet. Bei der Bearbeitung multikriterieller Optimierungsaufgaben können die Zielkriterien in einer Zielfunktion zusammengefasst oder es können mehrere voneinander unabhängige Zielfunktionen verwendet werden.

Deterministische Optimierungsverfahren können gemäß ihrer grundlegenden Formulierung ausschließlich eine Zielfunktion verarbeiten. Stochastische Optimierungsverfahren hingegen können eine oder mehrere Zielfunktionen berücksichtigen, z. B. multikriterielle genetische Algorithmen [Cav13]. Strategien der multikriteriellen Optimierung werden in Abschnitt 2.8.1 beschrieben.

- **Restriktionen:** Bei der Berücksichtigung von Restriktionen werden Optimierungen *ohne Restriktionen* und Optimierungen *mit Restriktionen* unterschieden. Die direkte Berücksichtigung von Restriktionen ist ausschließlich in einigen deterministischen Verfahren möglich (s. Abschnitt 2.6.1). Alternativ müssen Restriktionen über eine indirekte Formulierung berücksichtigt werden, z. B. über Straffunktionen. Soll sichergestellt werden, dass unter Verwendung eines deterministischen Optimierungsverfahrens das globale Optimum gefunden wird, müssen neben der Zielfunktion auch die Restriktionsfunktionen konvex sein (s. Abschnitt 2.2). Da die indirekte Formulierung der Restriktionen auch als multikriterielle Optimierung gesehen werden kann, werden in Abschnitt 2.8.1 Strategien zur Implementierung von Restriktionen in die Zielfunktion beschrieben.
- **Anzahl der verwendeten Disziplinen:** Gemäß der Anzahl der innerhalb des Evaluationsmodells verwendeten Disziplinen bzw. Fachbereiche wird die Optimierung als *monodisziplinär* oder *multidisziplinär* bezeichnet. Entscheidend hierfür sind die Submodelle des Evaluationsmodells. Multidisziplinäre Optimierungen verwenden bei der

Evaluation in der Regel verschiedene Modelle, welche teilweise oder vollständig parallel bearbeitet werden können.

Die aus den Evaluationen der Modelle ermittelten Zustandsvariablen bilden meist auch die Zielkriterien bzw. Restriktionen der Optimierung. Multidisziplinäre Optimierungsaufgaben stellen somit oft multikriterielle Probleme dar. Da für die multidisziplinäre Optimierung lediglich das Evaluationsmodell maßgebend ist, kann die Wahl des Optimierungsverfahrens unabhängig von der Multidisziplinarität, jedoch unter Berücksichtigung multikriterieller Optimierungsstrategien erfolgen. Strategien der multidisziplinären Optimierung werden in Abschnitt 2.8.2 beschrieben.

- **Kombination von Optimierungsmethoden und -verfahren:** Optimierungsmethoden und -verfahren können auf unterschiedliche Art miteinander kombiniert werden, wodurch die spezifischen Vorteile der jeweiligen Methoden und Verfahren genutzt werden können. Die Kombination kann *sequentiell*, *parallel* oder *integriert* (Multi-Level) erfolgen. Somit können Effektivität und Effizienz der Methoden bzw. Verfahren gesteigert werden. Reine Kombinationen von Optimierungsverfahren werden auch als hybride Optimierungsverfahren bezeichnet (s. Abschnitt 2.6.3). In Abschnitt 2.8.3 werden Strategien zur Kombination von Optimierungsmethoden und -verfahren beschrieben.

In der Produktentwicklung sind multikriterielle und multidisziplinäre Optimierungsstrategien von besonderer Bedeutung, da in der Regel mehrere Anforderungen an das zu entwickelnde Produkt berücksichtigt werden, welche zudem das Zusammenspiel verschiedener Disziplinen der Produktentwicklung erfordern. Diese Optimierungsstrategien werden daher in den folgenden Abschnitten detaillierter erläutert. Weiterhin wird die Kombination von Optimierungsmethoden und -verfahren gesondert beschrieben, da sich hierdurch die Effektivität und Effizienz der Methoden bzw. Verfahren steigern lässt.

2.8.1 Multikriterielle Optimierung

Multikriterielle Optimierungsaufgaben sind durch mehrere Zielkriterien charakterisiert. Hierbei wird auch von mehreren Zielfunktionen gesprochen, welche in dem Zielfunktionsvektor \mathbf{f} zusammengefasst werden können. Multikriterielle Optimierungen werden auch als Mehrzieloptimierungen, Vektor- oder Pareto-Optimierungen bezeichnet [Esc85], [Har14].

Die Lösung einer multikriteriellen Optimierungsaufgabe stellt im Gegensatz zu einer monokriteriellen Optimierung keinen Punkt im Designvariablen- bzw. Lösungsraum dar (s. Abschnitt 2.4), sondern wird durch eine Menge von Punkten gebildet. Diese Menge wird als Menge Pareto-optimaler Lösungen bzw. Pareto-Front P bezeichnet. Die Darstellung der Pareto-Front erfolgt durch Gegenüberstellung der Zielkriterien im Lösungsraum F . Dies ist exemplarisch für die Minimierung von zwei Zielkriterien (f_1, f_2) in Abbildung 2.37 dargestellt.

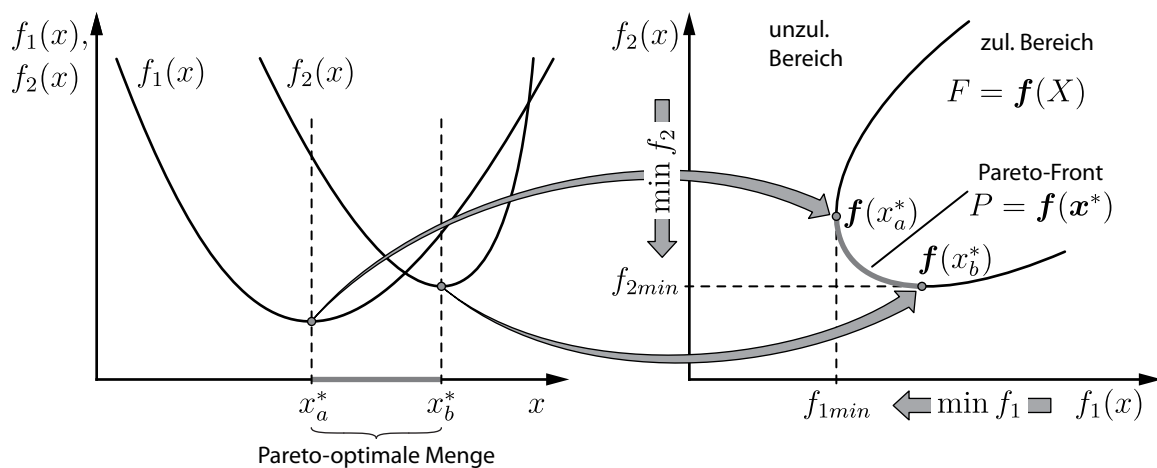


Abbildung 2.37: Abbildung des multikriteriellen Lösungsraumes aus den Zielkriterien, nach [Kin11], [Har14]

Die Pareto-Front P wird durch die Optima der beiden Zielkriterien f_1 und f_2 begrenzt, die sich aus den optimierten Designvariablen x_a^* und x_b^* ergeben. Diese Begrenzungspunkte werden im Lösungsraum durch die Punkte $\mathbf{f}(x_a^*)$ und $\mathbf{f}(x_b^*)$ repräsentiert, welche durch den Zielfunktionsvektor \mathbf{f} generiert werden. Aufgrund der Begrenzung der Pareto-Front durch die Optima der Zielkriterien ist innerhalb der Menge der Pareto-optimalen Lösungen die Verbesserung eines Zielkriteriums nur möglich, wenn sich dabei ein anderes Zielkriterium verschlechtert. Dieser Zustand wird auch als Pareto-Optimalität bezeichnet [Har14]. Da aufgrund der Pareto-Optimalität gemäß der Zielkriterien keine besseren Lösungen ermittelt werden können, liegt die Pareto-Front stets am Rand des Lösungsraumes. Daher wird auch vom Pareto-optimalen Rand gesprochen [Sch13].

Die Ermittlung der Pareto-Front ist nicht auf die reine Minimierung der Zielkriterien beschränkt, sondern kann analog auch bei der Maximierung einzelner oder aller Zielkriterien verwendet werden. Auch hierbei liegt die Pareto-Front am Rand des Lösungsraumes, wie in Abbildung 2.38 dargestellt ist.

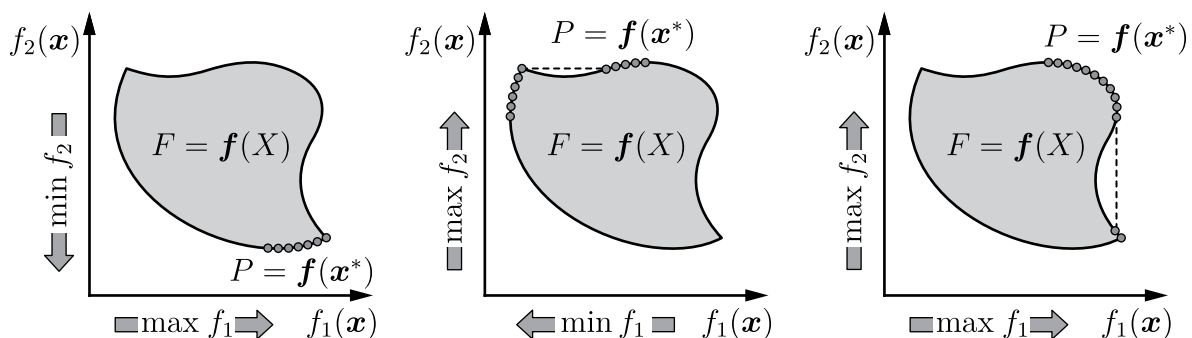


Abbildung 2.38: Pareto-Fronten im Lösungsraum in Abhängigkeit vom Optimierungsziel, nach [Bes06]

Wie in den beiden rechten Diagrammen dargestellt, ist die Pareto-Front nicht zwangsläufig zusammenhängend, sondern kann auch aus mehreren Teilstücken bestehen. Dies tritt auf, wenn der Lösungsraum F nicht konvex ist (s. Abschnitt 2.2). In den nicht konvexen Bereichen (s. Strichlinie in Abbildung 2.38) können keine Pareto-optimalen Lösungen

gefunden werden. Neben nicht konvexen Pareto-Fronten erschweren auch lokale und unstetige Pareto-Fronten die Ermittlung der Pareto-optimalen Lösungen [Bes06]. Diese drei Sonderfälle sind in Abbildung 2.39 dargestellt.

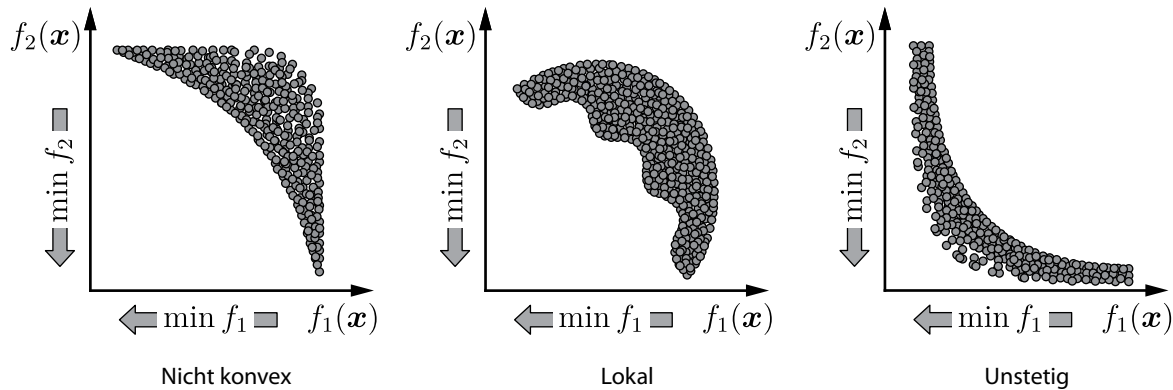


Abbildung 2.39: Charakteristische Pareto-Fronten im Lösungsraum, nach [Bes06]

In den Diagrammen sind jeweils die Lösungen des vollständigen Lösungsraumes dargestellt, um somit den gesamten Verlauf der Pareto-Fronten sichtbar zu machen. Die Ermittlung der Lösungen kann z. B. durch zufallsbasierte explorative Untersuchung des Lösungsraumes erfolgen (s. Abschnitt 2.7.1). Lokale Pareto-Fronten sind durch konvexe Bereiche charakterisiert, welche durch nicht konvexe Bereiche unterbrochen sind. Sind diese konvexen Bereiche so klein, dass nur eine oder wenige Pareto-optimale Lösungen ein Teilgebiet der gesamten Pareto-Front bilden, wird die Pareto-Front als unstetig bezeichnet.

Da die Maximierung eines Zielkriteriums der Minimierung seines Negativen entspricht, beschränken sich die weiteren Ausführungen dieses Abschnittes auf die Minimierung der Zielkriterien

Bisher wurden ausschließlich zweidimensionale Pareto-Fronten betrachtet. Je nach der Anzahl der Zielkriterien kann eine Pareto-Front verschiedene Dimensionen aufweisen. Die Dimensionalität der Pareto-Front entspricht hierbei der Dimensionalität bzw. der Anzahl der Zielkriterien. So führen drei Zielkriterien zu einer dreidimensionalen Pareto-Front. Diese entspricht einer im Raum liegenden Fläche mit drei Eckpunkten, welche die Optima der einzelnen Zielkriterien repräsentieren. Höherdimensionale Probleme führen zu höherdimensionalen Pareto-Fronten. Insbesondere bei höherdimensionalen Optimierungsaufgaben ist eine formale Beschreibung der Pareto-Optimalität notwendig, da die Lösungen dieser Probleme nicht mehr grafisch gegenübergestellt werden können.

Zur Beurteilung der Pareto-Optimalität verschiedener Lösungen wird der Begriff der Dominanz verwendet. Eine Lösung \mathbf{x}_a dominiert eine andere Lösung \mathbf{x}_b wenn mindestens ein Zielfunktionswert der Lösung \mathbf{x}_a besser ist als der Wert der gleichen Zielfunktion von \mathbf{x}_b und dabei alle weiteren Zielfunktionswerte mindestens genau so gut sind [Gol89]. Maßgebend hierfür sind die Dominanzzone sowie die dominierte Zone einer Lösung bzw. eines Punktes im Lösungsraum. Diese sind für einen zweidimensionalen Lösungsraum in Abbildung 2.40 dargestellt.

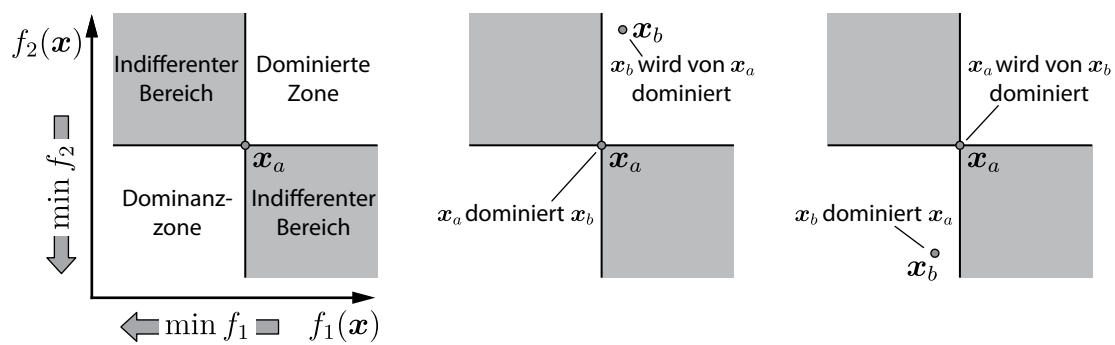


Abbildung 2.40: Dominanz von Lösungen im Lösungsraum, nach [Har14]

Die Dominanzzone des Punktes \mathbf{x}_a enthält alle Punkte, welche den Punkt \mathbf{x}_a dominieren. Analog dazu beinhaltet die dominierte Zone des Punktes alle Punkte, die von diesem dominiert werden. In den indifferenten Bereichen ist keine Aussage über eine Dominanz möglich [Har14]. Wird ein Punkt von keinem anderen Punkt dominiert, wird dieser als *nicht dominiert* bezeichnet und gilt als Pareto-optimal [Gol89]. Diese Einteilung ist vor allem für multikriterielle Optimierungsverfahren relevant, welche im weiteren Verlauf dieses Abschnitts beschrieben werden. Die Menge Pareto-optimaler Lösungen besteht somit ausschließlich aus Lösungen, welche von keiner anderen Lösung dominiert werden [Har14].

Zur Bewertung von Lösungen in der Produktentwicklung lassen sich verschiedene Arten von Zielkriterien identifizieren, welche auch in der Natur wiedergefunden werden können. Folgende Arten von Zielkriterien werden unterschieden [VK10]:

- **KO-Kriterien:** Diese Kriterien müssen in jedem Fall durch eine Lösung erfüllt werden. Die Nichterfüllung der Kriterien resultiert in einem Ausschluss oder einer sehr schlechten Bewertung. KO-Kriterien sind primär bei der Restriktionsformulierung oder der manuellen Auswahl geeigneter Lösungen auf Basis der Pareto-optimalen Menge relevant. Beispiele sind die Vermeidung bestimmter Eigenfrequenzen, die Einhaltung zulässiger Spannungen und die Gewährleistung einer definierten Lebensdauer.
- **Attraktivitätskriterien:** Zielkriterien dieser Art sollen möglichst gering oder möglichst hoch sein. Die Auswahl geeigneter Lösungen erfolgt manuell aus der Menge der Pareto-optimalen Lösungen. Beispiele sind die Minimierung der Masse und die Maximierung der Steifigkeit.
- **Attraktivitätskriterien mit Schranke:** Diese Zielkriterien stellen eine Kombination der beiden ersten dar. Sie beinhalten einen Schwellwert, welcher mindestens erreicht werden muss, und sollen zusätzlich möglichst gering oder möglichst hoch sein. Diese Art von Zielkriterien ist in der Produktentwicklung sehr oft zu finden. Die Auswahl geeigneter Lösungen erfolgt manuell aus der Menge der Pareto-optimalen Lösungen. Zusätzlich können Restriktionen formuliert werden. Beispiele sind die Vermeidung bestimmter Eigenfrequenzen bei möglichst hohen Eigenfrequenzen oder die Einhaltung zulässiger Spannungen bei gleichzeitiger Minimierung.

Auf Basis dieser Arten von Zielkriterien erfolgt die Festlegung der multidisziplinären Optimierungsstrategie. Dabei haben sich verschiedene Strategien etabliert. Neben diesen existiert eine Vielzahl weiterer multidisziplinärer Optimierungsstrategien, welche oft Speziallösungen bezogen auf das jeweilige Optimierungsproblem darstellen. Die Auswahl

einer multidisziplinären Optimierungsstrategie muss zudem unter Berücksichtigung des Optimierungsverfahrens erfolgen. Gemäß ihrer Definition können deterministische Optimierungsverfahren nur eine Zielfunktion berücksichtigen. Stochastische Optimierungsverfahren hingegen können eine oder mehrere Zielfunktionen verarbeiten, da diese keine Suchrichtungen oder Ableitungen verwenden [Cav13].

Zur Verwendung von monokriteriellen Optimierungsverfahren bei multikriteriellen Problemen, muss eine Skalarisierung der vektoriellen Zielfunktion erfolgen [Bes06]. Dabei werden die verschiedenen Zielkriterien in einer Zielfunktion zusammengefasst, welche auch als Ersatzzielfunktion bezeichnet wird [Har14]. Die Formulierung dieser Zielfunktion kann durch Zielgewichtung, die Verwendung einer Abstandsfunktion oder die Formulierung von Restriktionen erfolgen. Da das Ziel der Optimierung somit ein konkreter Wert ist, welcher im Lösungsraum durch einen Punkt repräsentiert wird, ist mit diesen Optimierungsstrategien nur die Ermittlung eines Punktes bzw. eines bestimmten Ausschnitts der Pareto-Front möglich. Zur Bestimmung der Pareto-Front müssen mehrere Optimierungsläufe mit unterschiedlichen Einstellungen durchgeführt werden. Da bei der Skalarisierung der vektoriellen Zielfunktion keine Veränderung des Optimierungsverfahrens erfolgt, können diese Strategien auch bei Optimierungsmethoden monokriterieller Optimierungsverfahren angewendet werden, z. B. bei der Topologieoptimierung [ZZF14].

Alternativ hierzu können multikriterielle stochastische Optimierungsverfahren verwendet werden, bei denen die direkte Berücksichtigung mehrerer Zielkriterien erfolgt, z. B. multikriterielle genetische Algorithmen. Dies hat den Vorteil, dass die Pareto-Front in ihrer gesamten Breite ermittelt wird, womit jedoch auch eine größere Anzahl an Evaluationen verbunden ist.

Die genannten Strategien werden im Folgenden erläutert. Für weitere Informationen zu multidisziplinären Optimierungsstrategien wird auf [Esc85], [MA04], [Bes06], [Sch13], [Har14] verwiesen.

2.8.1.1 Zielgewichtung

Die Zielgewichtung gehört zu den einfachsten und am meisten verwendeten multidisziplinären Optimierungsstrategien [Bes06]. Die einzelnen Zielkriterien werden hierbei gewichtet und aufsummiert. Daher wird auch von gewichteten Zielkriterien bzw. gewichteten Zielfunktionen gesprochen. Die Formulierung der Zielfunktion erfolgt nach der folgenden Gleichung [Har14]:

$$f(\mathbf{x}) = \sum_{i=1}^{n_f} w_i \cdot f_i(\mathbf{x}) \quad \text{mit} \quad w \in \mathbb{R}^+ \quad (2.22)$$

Die Gewichtung der Zielkriterien f_i basiert auf Gewichtungsfaktoren w_i , welche positive reelle Zahlen darstellen. Die Variable n_f repräsentiert die Anzahl der Zielkriterien. Über die Gewichtungsfaktoren kann der Einfluss der jeweiligen Zielkriterien und somit der Zielpunkt auf der Pareto-Front gesteuert werden. Die geometrische Repräsentation dieser Optimierungsstrategie ist exemplarisch für die Minimierung zweier Zielkriterien (f_1, f_2) in Abbildung 2.41 dargestellt.

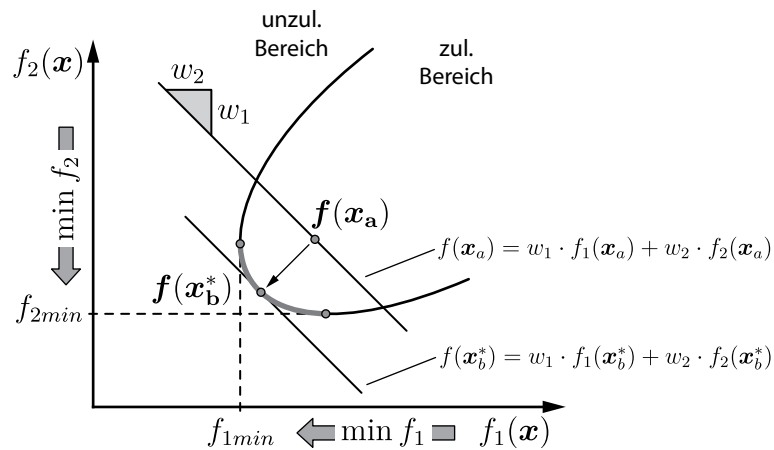


Abbildung 2.41: Formulierung der Zielfunktion durch Zielgewichtung, nach [Bes06], [Har14]

Ausgehend von einem Punkt $\mathbf{f}(\mathbf{x}_a)$ im Lösungsraum wird das dazugehörige Skalar $f(\mathbf{x}_a)$ bestimmt, welches eine Gerade im Lösungsraum darstellt. Im Verlauf der Optimierung wird diese Gerade parallel in Richtung des Minimums verschoben. Ist der minimale skalare Wert der Zielfunktion erreicht (hier $f(\mathbf{x}_b^*)$), bildet die dazugehörige Gerade die Tangente der Pareto-Front im Punkt $\mathbf{f}(\mathbf{x}_b^*)$. Durch die Variation der Gewichtungsfaktoren (w_1 , w_2) kann die Steigung der Geraden verändert werden, wodurch der Lösungspunkt auf der Pareto-Front verschoben und somit der Einfluss der jeweiligen Zielkriterien gesteuert wird [Bes06].

Zur Ermittlung einer geeigneten Lösung müssen sinnvolle Gewichtungsfaktoren verwendet werden. Diese können geschätzt werden, wenn bereits ausreichende Erfahrungen über das Verhalten des zu optimierenden Objekts vorliegen [VWBZ09]. Die Optimierung kann somit über die Gewichtungsfaktoren auf einen bestimmten Bereich der Pareto-Front fokussiert werden. Dieser Bereich muss jedoch bekannt sein, z. B. bei einer bereits bekannten Optimierungsaufgabe.

Ist dieser Bereich nicht bekannt, ist die Bestimmung der Gewichtungsfaktoren schwierig, z. B. bei einer vollständig neuen Optimierungsaufgabe. Liegen die Zielkriterien zudem in unterschiedlichen Dimensionen vor, wird die Bestimmung der Gewichtungsfaktoren zusätzlich erschwert. Durch die Normierung der Zielkriterien, z. B. auf ihre abgeschätzten Maximalwerte, kann dieser Nachteil reduziert werden und die Gewichtungsfaktoren gemäß der Wichtigkeit der Zielkriterien angenommen werden. Diese Strategie wird in dieser Arbeit u. a. bei der Optimierung in Fallstudie 3 verwendet (s. Anhang A.7.2). Liegen keine Präferenzen der Zielkriterien vor, ist es in einem ersten Optimierungslauf sinnvoll, die Gewichtungsfaktoren, so zu wählen, dass die Summanden der Zielfunktion zunächst etwa gleich groß sind [Har14].

Ein weiterer Nachteil der Zielgewichtung ist, dass dem Optimierungsalgorithmus durch die Gewichtungsfaktoren eine bestimmte Suchrichtung aufgezwungen wird [VWBZ09]. Die gesamte Pareto-Front lässt sich nur durch mehrere Optimierungsläufe mit unterschiedlichen Gewichtungsfaktoren bestimmen. Hierbei führt eine gleichmäßige Verteilung der Gewichtungsfaktoren jedoch nicht zwangsläufig zu einer gleichmäßigen Verteilung der Lösungspunkte auf der Pareto-Front. In den Bereichen, in denen sich die Steigung der Pareto-Front stark ändert, werden mehrere Lösungspunkte benötigt als in Bereichen schwacher

Steigungsänderung [Har14]. Eine gleichmäßige Verteilung der Gewichtungsfaktoren bietet jedoch eine gute Basis für weiterführende Untersuchungen der Pareto-Front des Optimierungsproblems. Da die verschiedenen Optimierungsläufe unabhängig voneinander sind, können diese parallel durchgeführt werden.

Der größte Nachteil diese Strategie kommt bei nicht konvexen Pareto-Fronten zum Tragen. Da die Zielfunktion durch Geraden repräsentiert wird, welche bei einer Optimierung parallel verlaufen, können die nicht konvexen Bereiche der Pareto-Fronten nicht gefunden werden. Diese Bereiche werden übergangen [Bes06]. In diesem Fall sollte eine andere Optimierungsstrategie verwendet werden.

2.8.1.2 Abstandsfunktion

Durch die Formulierung einer Abstandsfunktion können die Lösungspunkte der Optimierung gezielter beeinflusst und auch nicht konvexe Bereiche gefunden werden. Hierzu wird für jedes Zielkriterium f_i ein Anspruchsniveau \bar{y}_i definiert, welches theoretisch durch die Optimierung erreicht werden soll. Das Anspruchsniveau stellt das ideale Ziel der Optimierung dar, das z. B. aufgrund von Restriktionen nicht erreicht werden kann, jedoch angestrebt wird. Das Ziel der Optimierung ist es, den Abstand zu diesem Niveau zu minimieren. Dabei werden die Differenzen zwischen den Zielkriterien und deren Anspruchsniveaus mittels der folgenden Zielfunktion aufsummiert [Esc85], [Har14]:

$$f(\mathbf{x}) = \left(\sum_{i=1}^{n_f} |f_i(\mathbf{x}) - \bar{y}_i|^p \right)^{\frac{1}{p}} \quad \text{mit } 1 \leq p < \infty \quad (2.23)$$

Diese Form der Summierung wird auch als p -Norm bezeichnet. Durch die Variation des Faktors p können aus der p -Norm verschiedene Normen abgeleitet werden, welche verschiedene Abstandsdefinitionen, sog. Metriken, darstellen. Zur multikriteriellen Optimierung wird oft eine der folgenden Normen verwendet [Alt12]:

$$\text{Summennorm } p = 1 : \quad f(\mathbf{x}) = \sum_{i=1}^{n_f} |f_i(\mathbf{x}) - \bar{y}_i| \quad (2.24)$$

$$\text{Euklidische Norm } p = 2 : \quad f(\mathbf{x}) = \sqrt{\sum_{i=1}^{n_f} |f_i(\mathbf{x}) - \bar{y}_i|^2} \quad (2.25)$$

$$\text{Maximumnorm } p \rightarrow \infty : \quad f(\mathbf{x}) = \max(|f_i(\mathbf{x}) - \bar{y}_i|) \quad (2.26)$$

Die einfachste Form der Abstandsfunktion stellt die *Summennorm* dar. Hierbei werden die Abstände zwischen den Zielkriterien und ihren Anspruchsniveaus summiert. Dieser Wert wird im Verlauf der Optimierung minimiert. In einem zweidimensionalen Lösungsraum entspricht dies einer Raute, bei der alle Winkel gleich groß sind. Der Mittelpunkt der Raute repräsentiert den Vektor der Anspruchsniveaus $\bar{\mathbf{y}}$, welcher die Anspruchsniveaus der einzelnen Zielkriterien enthält und den Zielpunkt der Optimierung darstellt. Da die Kanten der Raute in einem Winkel von 45° verlaufen, ist die Summennorm analog zur Zielgewichtung zu sehen, bei der die Gewichtungsfaktoren gleich groß sind. Der Unterschied besteht jedoch darin, dass bei der Summennorm ausschließlich der Anspruchsniveauektor $\bar{\mathbf{y}}$ als Ziel der Optimierung vorgegeben wird und keine Abschätzung von Gewichtungsfaktoren erfolgt.

Die *Euklidische Norm* verwendet die euklidische Abstandsdefinition. Im zweidimensionalen Lösungsraum wird der Abstand zwischen dem Anspruchsniveauektor $\bar{\mathbf{y}}$ und einem Lösungspunkt mittels eines Kreises bestimmt, dessen Mittelpunkt in $\bar{\mathbf{y}}$ liegt. Da mittels der euklidischen Norm somit auch Lösungspunkte nicht konvexer Pareto-Fronten bestimmt werden können, findet diese sehr oft Anwendung [Ehr05], [Bes06].

Bei der *Maximumnorm* wird ausschließlich das Zielkriterium berücksichtigt, welches aktuell den größten Abstand zum Anspruchsniveau aufweist [Ehr05]. Diese Norm wird auch als Tschebyschew-Norm bezeichnet [BSG⁺13].

Neben der Wahl des Faktors p ist die Definition der Anspruchsniveaus ausschlaggebend für das Optimierungsergebnis. Der Einfluss des Anspruchsniveauektors $\bar{\mathbf{y}}$ auf das Optimierungsergebnis $\mathbf{f}(\mathbf{x}^*)$ ist in Abbildung 2.42 für drei Optimierungen (a, b, c) dargestellt. Das Ziel der Optimierung ist die Minimierung der Zielkriterien f_1 und f_2 .

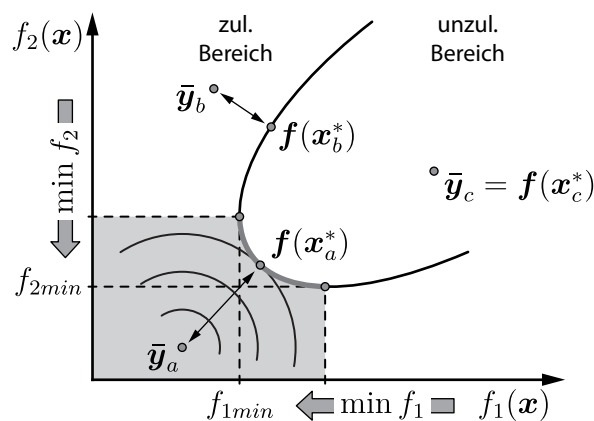


Abbildung 2.42: Lösungen bei verschiedenen Anspruchsniveaus, nach [Esc85], [Bes06]

Die Ermittlung des Abstände erfolgt hierbei mit der Euklidischen Norm ($p = 2$). Wie exemplarisch für den Anspruchsniveauektor $\bar{\mathbf{y}}_a$ dargestellt ist, wird der Abstand zwischen Lösungspunkt und Anspruchsniveauektor durch Kreise um den Anspruchsniveauektor ermittelt. Der Lösungspunkt, welcher den geringsten Abstand zum Anspruchsniveau aufweist, repräsentiert einen Punkt der Pareto-Front (hier $\mathbf{f}(\mathbf{x}_a^*)$). Wird das Anspruchsniveau ungünstig gewählt ($\bar{\mathbf{y}}_b$), liegt der optimierte Lösungspunkt unter Umständen nicht auf der Pareto-Front ($\mathbf{f}(\mathbf{x}_b^*)$). Bei der Definition des Anspruchsniveaus innerhalb des zulässigen Bereiches des Lösungsraumes ($\bar{\mathbf{y}}_c$), kann der Zielpunkt durch die Optimierung erreicht werden ($\mathbf{f}(\mathbf{x}_c^*)$). In diesem Fall liegt der ermittelte Lösungspunkt ebenfalls nicht auf der Pareto-Front und die Optimierung sollte unter Verwendung eines neuen Anspruchsniveaus wiederholt werden. Sind die Optima der einzelnen Zielfunktionen bekannt bzw. können diese abgeschätzt werden, kann das Anspruchsniveau ausgehend von diesen Werten festgelegt werden (s. grauer Bereich in Abbildung 2.42). Alternativ zur reinen Abschätzung der Anspruchsniveaus kann zunächst für jedes Zielkriterium eine monokriterielle Optimierung durchgeführt werden. Die Ergebnisse dieser Optimierung werden anschließend als Anspruchsniveau der multikriteriellen Optimierung verwendet. Der Nachteil bei dieser Vorgehensweise ist jedoch die hohe Anzahl an Evaluationen aufgrund der vielen Optimierungen [Har14].

Werden Zielkriterien unterschiedlicher Dimensionen verwendet, sollten diese in der Zielfunktion auf ihr Anspruchsniveau normiert werden, da sonst eine unausgeglichene Gewichtung innerhalb der Zielfunktion entsteht und Zielkriterien mit größeren Werten einen

größeren Einfluss auf den Zielfunktionswert haben. Die Normierung erfolgt gemäß der folgenden Zielfunktion [Har14]:

$$f(\mathbf{x}) = \left(\sum_{i=1}^{n_f} \frac{|f_i(\mathbf{x}) - \bar{y}_i|^p}{|\bar{y}_i|} \right)^{\frac{1}{p}} \quad \text{mit } 1 \leq p < \infty \quad (2.27)$$

Zusätzlich können in der Zielfunktion für jedes Zielkriterium Gewichtungsfaktoren verwendet werden, welche mit den Differenzen der jeweiligen Zielkriterien multipliziert werden. Somit lässt sich ähnlich der Zielgewichtung weiterer Einfluss auf das Optimierungsergebnis nehmen [Ehr05].

Abstandsfunktionen bieten gegenüber der reinen Zielgewichtung zwei entscheidende Vorteile. So lassen sich im Vorfeld einer Optimierung die Anspruchsniveaus einfacher abschätzen als die Gewichtungsfaktoren, insbesondere bei einer neuen, unbekanntem Optimierungsaufgabe. Durch Variation der Anspruchsniveaus können in mehreren Optimierungsläufen verschiedene Punkte der Pareto-Front ermittelt werden [Bes06]. Analog zur Zielgewichtung ist die parallele Durchführung dieser Optimierungsläufe möglich. Zudem können aufgrund der Definition der Abstandsfunktion auch Lösungspunkte nicht konvexer Pareto-Fronten ermittelt werden, z. B. mittels Euklidischer Norm [Ehr05]. Werden bei einer stark konkaven Krümmung der Pareto-Front, einige Lösungspunkte nicht gefunden, kann durch die Erhöhung des Faktors p dieses Verhalten verbessert werden [Bes06].

2.8.1.3 Restriktionsformulierung

Bei der Restriktionsformulierung findet ebenfalls eine Skalarisierung des multikriteriellen Optimierungsproblems statt. Die einzelnen Zielkriterien werden hierbei nicht in einer Zielfunktion zusammengefasst, sondern es wird ein Zielkriterium minimiert bzw. maximiert, während die anderen Zielkriterien als Restriktionen berücksichtigt werden. Da hierbei eine Transformation der eigentlichen Optimierungsaufgabe stattfindet, wird dies auch als *restriktionsorientierte Transformation* bezeichnet [Esc85], [Sch13]. Das Optimierungsproblem wird anschließend durch die folgende Gleichung beschrieben [Har14]:

$$\min f_j(\mathbf{x}) \quad \text{mit } f_i(\mathbf{x}) \leq f_i^u ; \quad i = 1, n_f ; \quad i \neq j \quad (2.28)$$

Es findet somit eine Minimierung des Zielkriteriums $f_j(\mathbf{x})$ statt. Die weiteren Zielkriterien $f_i(\mathbf{x})$ werden als Restriktionen betrachtet, wobei in der Regel ein oberer Grenzwert f_i^u verwendet wird [Bes06]. Die Definition unterer Grenzwerte ist ebenfalls möglich. Bei der Formulierung der Optimierungsaufgabe muss der Anwender entscheiden, welches Kriterium das wichtigste darstellt. Dieses Zielkriterium wird minimiert. Für die restlichen Zielkriterien werden Grenzwerte definiert, welche in jedem Fall im zulässigen Lösungsraum liegen müssen.

Die geometrische Repräsentation der Restriktionsformulierung ist exemplarisch für zwei Zielkriterien (f_1, f_2) in Abbildung 2.43 dargestellt. Dabei wird das Zielkriterium f_2 minimiert und das Zielkriterium f_1 als Restriktion verwendet, wobei f_1^u den oberen Grenzwert darstellt.

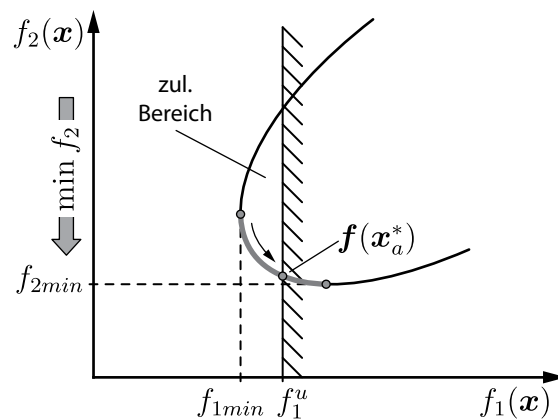


Abbildung 2.43: Restriktionsformulierung der Zielfunktion, nach [Bes06], [Har14]

Durch die Verwendung der Restriktion wird der zulässige Bereich des Lösungsraumes verkleinert und die Optimierung tendiert zum minimalen Wert des Zielkriteriums f_2 innerhalb des zulässigen Lösungsraumes. Die optimale Lösung $\mathbf{f}(\mathbf{x}_a^*)$ bildet den Schnittpunkt zwischen der Pareto-Front und der Restriktion. Werden in verschiedenen Optimierungsläufen unterschiedliche Grenzwerte verwendet, können weitere Punkte der Pareto-Front ermittelt werden [Bes06]. Diese Optimierungsläufe können parallel durchgeführt werden.

Aufgrund der Restriktionsformulierung können vor allem deterministische Optimierungsverfahren zur Lösung multikriterieller Optimierungsprobleme genutzt werden, bei denen Restriktionen direkt berücksichtigt werden (s. Abschnitt 2.6.1). Ist die direkte Berücksichtigung der Restriktionen nicht möglich, lassen sich diese indirekt verwenden, indem das Optimierungsproblem mit Restriktionen in ein Ersatzproblem ohne Restriktionen überführt wird. Somit können auch stochastische Optimierungsverfahren eingesetzt werden (s. Abschnitt 2.6.2). Dabei hat sich die Verwendung von Straffunktionen (engl. *penalty functions*) etabliert [Ulb06]. Das Ersatzproblem wird dadurch wie folgt formuliert [Har14]:

$$\min \tilde{f}(\mathbf{x}) \quad \text{mit} \quad \tilde{f}(\mathbf{x}) = f(\mathbf{x}) + q \cdot p(\mathbf{x}) \quad (2.29)$$

Die zur Optimierung verwendete Ersatzzielfunktion $\tilde{f}(\mathbf{x})$ besteht aus der skalaren monokriteriellen Zielfunktion $f(\mathbf{x})$ und der Straffunktion $p(\mathbf{x})$, deren Steigung zusätzlich über den Skalierungsfaktor q gesteuert werden kann. Die Straffunktion einer gemäß der Gleichungen (2.1) – (2.3) definierten Optimierungsaufgabe kann durch eine der folgenden Formen beschrieben werden [Har14]:

$$\text{Externe Straffunktion:} \quad p(\mathbf{x}) = \sum_{j=1}^{n_g} [\max(g_j(\mathbf{x}), 0)]^2 + \sum_{k=1}^{n_h} [h_k(\mathbf{x})]^2 \quad (2.30)$$

$$\text{Interne Straffunktion:} \quad p(\mathbf{x}) = \sum_{j=1}^{n_g} \frac{-1}{g_j(\mathbf{x})} \quad (2.31)$$

Externe Straffunktionen können Ungleichheitsrestriktionen $g_j(\mathbf{x})$ und Gleichheitsrestriktionen $h_k(\mathbf{x})$ beinhalten. Bei der Definition einer Ungleichheitsrestriktion liefert die Straffunktion erst einen Wert größer 0, wenn die Restriktion verletzt wird und somit der Wert von g ebenfalls größer 0 ist. Durch das Quadrieren des Terms verläuft die Straffunktion exponentiell zur Größe der Überschreitung der Restriktion. Die Verletzung einer Gleichheitsrestriktion resultiert bei jeglicher Abweichung in einem Strafterm größer 0. Quadratische

Strafterme stellen die am meisten verwendete Form dar. Auch die Verwendung höhergradiger Strafterme ist möglich [Ul06]. Bei internen Straffunktionen können ausschließlich Ungleichheitsrestriktionen verwendet werden, da die Restriktion eine Asymptote der Straffunktion darstellt und sich die Straffunktion dieser Restriktion annähert. Liegen die Restriktionen nicht in der Form der Gleichungen (2.2) und (2.3) vor, müssen diese zur Verwendung in der Straffunktion umgeformt werden. Abbildung 2.44 zeigt die grafische Darstellung beider Straffunktionsformulierungen für eine Ungleichheitsrestriktion.

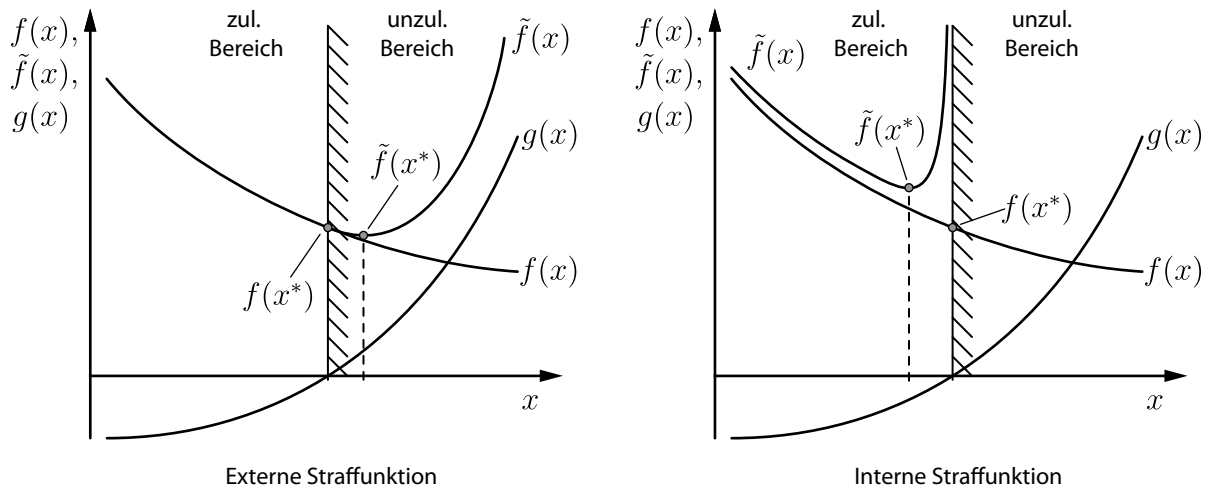


Abbildung 2.44: Externe und interne Straffunktionen, nach [Har14]

Die externe Straffunktion wird erst im unzulässigen Bereich hinter der Restriktion wirksam, was zu einem starken Anstieg der Ersatzzielfunktion $\tilde{f}(x)$ führt. Im zulässigen Bereich sind Ziel- und Ersatzzielfunktion identisch. Die Grenze, welche den zulässigen und den unzulässigen Bereich voneinander trennt, resultiert aus der Ungleichheitsrestriktion $g(x)$. Ist der Wertebereich der Ungleichheitsrestriktion negativ, sind die Lösungen in diesem Bereich zulässig. Lösungen im positiven Wertebereich von $g(x)$ sind unzulässig. Die interne Straffunktion ist bereits im zulässigen Bereich wirksam, was dazu führt, dass Ziel- und Ersatzzielfunktion voneinander abweichen und sich die Ersatzzielfunktion der Grenze zum unzulässigen Bereich annähert.

Bei beiden Straffunktionen weicht das Minimum der Ersatzzielfunktion $\tilde{f}(x^*)$ vom Minimum der Zielfunktion $f(x^*)$ ab. Diese Abweichung kann bei externen Straffunktionen durch eine Erhöhung bzw. bei internen durch eine Verringerung des Skalierungsfaktors q reduziert werden, da somit die Steigung der Funktion erhöht wird. Zu große bzw. kleine Werte des Skalierungsfaktors führen bei der Optimierung jedoch zu numerischen Problemen, insbesondere, wenn der Startpunkt der Optimierung im unzulässigen Bereich liegt [Sch13].

Die beschriebenen Nachteile können durch eine schrittweise Anpassung des Skalierungsfaktors vermieden werden. Dies kann manuell erfolgen, indem zunächst eine Optimierung mit einem unkritischen Skalierungsfaktor durchgeführt wird und ausgehend von der gefundenen Lösung das exakte Optimum mit einem höheren bzw. niedrigeren Skalierungsfaktor in einer weiteren Optimierung ermittelt wird [Har14]. Alternativ hierzu kann die Anpassung des Skalierungsfaktors während der Optimierung bei Verwendung einer externen Straffunktion erfolgen. Dynamische Straffunktionen erhöhen den Skalierungsfaktor hierzu

mit zunehmender Anzahl der Iterationen. Bei adaptiven Straffunktionen wird der Skalierungsfaktor in Abhängigkeit der Lage der aktuell besten Lösung variiert. Unzulässige Lösungen führen zu einer Erhöhung des Skalierungsfaktors, zulässige Lösungen zu einer Verringerung [SC97].

In multikriteriellen Optimierungsaufgaben kommen in der Regel externe Straffunktionen zum Einsatz. Dabei können die Nachteile vernachlässigt werden, da eine geringe Überschreitung der Restriktion nicht problematisch ist, sondern lediglich einen anderen Lösungspunkt auf der Pareto-Front zur Folge hat. Zudem sollten bei der Verwendung genetischer Algorithmen geringe Überschreitungen der Restriktionen nur wenig bestraft werden, da diese Individuen im Verlauf der Optimierung ebenfalls zu guten Lösungen führen können. Eine externe Straffunktion wird in dieser Arbeit bei der Optimierung in Fallstudie 3 verwendet (s. Anhang A.7.2).

2.8.1.4 Multikriterielle genetische Algorithmen

Im Gegensatz zu den bisher beschriebenen multikriteriellen Optimierungsstrategien werden bei multikriteriellen genetischen Algorithmen (GA) alle Zielkriterien gleichermaßen bei der Fitnesszuweisung bzw. der Selektion der Individuen (s. Abschnitt 2.6.2.1) berücksichtigt. Eine Skalarisierung des Vektors der Zielkriterien findet somit nicht statt. Dies hat den Vorteil, dass die Pareto-Front in ihrer gesamten Breite ermittelt wird, was jedoch auch zu einer größeren Anzahl an Evaluationen führt. Multikriterielle GA werden auch als Pareto-basierte GA bezeichnet.

Vor allem GA sind für die Bearbeitung multikriterieller Optimierungsaufgaben geeignet, da diese auf Crossover-Operationen basieren [Cav13]. Durch die Crossover-Operationen werden zielführende Eigenschaften bzw. Designvariablenwerte bezogen auf einzelne Zielkriterien miteinander kombiniert und somit die Pareto-optimalen Eigenschaften bezogen auf alle Zielkriterien generiert.

Bei der Zuweisung der Fitnesswerte der Individuen hat sich die rangbasierte Fitnesszuweisung gemäß der Sortierung nicht dominierter Lösungen (engl. *nondominated sorting*) etabliert [Zit99]. Der Rang r_I eines Individuums (s. Abschnitt 2.6.2.1) wird dabei entsprechend dessen Dominanz festgelegt, da somit die Pareto-Optimalität des Individuums beschrieben wird (s. Abbildung 2.40). Die Individuen der Population, welche nicht von anderen Individuen dominiert werden, gelten in der aktuellen Population als Pareto-optimal und erhalten den Rang $r_I = 1$. Diese Individuen werden aus der Gruppe der zu bewertenden Individuen entfernt und es werden erneut die Individuen ermittelt, die nicht von anderen Individuen dominiert werden. Diese erhalten den Rang $r_I = 2$. Der Prozess wird wiederholt, bis allen Individuen der Population ein Rang zugewiesen wurde [Gol89]. Anschließend erfolgt die Zuweisung der Fitnesswerte gemäß des Ranges. Abbildung 2.45 zeigt die grafische Darstellung der Rangzuweisung. Der Rang eines Individuums ist dabei direkt neben dem Individuum dargestellt.

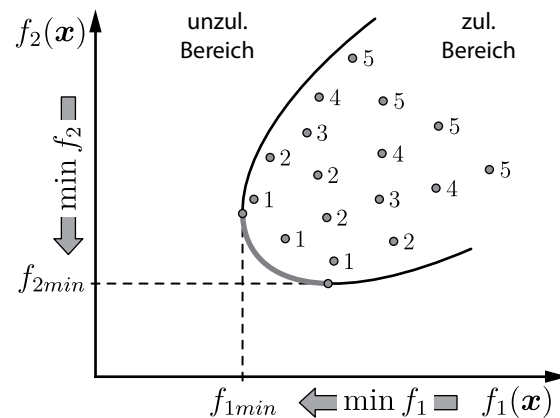


Abbildung 2.45: Rangzuweisung durch nicht dominierte Sortierung, nach [Bes06], [Har14]

Nach Ende der Optimierung wird die Pareto-Front durch alle Individuen repräsentiert, welche von keinen anderen Individuen dominiert werden und somit wiederum zu einem Rang von $r_I = 1$ führen würden. Da diese Individuen den gleichen Rang besitzen, sind sie gleichwertig, aufgrund der unterschiedlichen Werte der Designvariablen jedoch nicht gleichartig.

Das beschriebene Verfahren der Rangzuweisung durch nicht dominierte Sortierung wird aufgrund der Einfachheit sowie der hohen Effektivität sehr oft angewendet [Zit99]. Die Umsetzung erfolgt u. a. im Algorithmus NSGA-II (Nondominated Sorting Genetic Algorithm). Bei diesem Algorithmus werden die Individuen des gleichen Ranges zusätzlich gemäß ihrer Verteilung im Lösungsraum unterschieden, um eine möglichst gleichmäßige Verteilung der Population entlang der Pareto-Front zu gewährleisten [DPAM00], [DPAM02], [Kan08]. Da der Algorithmus auch zur Lösung monokriterieller Optimierungsaufgaben verwendet werden kann, wird dieser in der vorliegenden Arbeit bei der Optimierung der Prozessprioritäten eingesetzt (s. Abschnitt 5.4.2). Eine Übersicht etablierter, multikriterieller GA ist in Tabelle A.2 im Anhang A.2 zu finden. Ein detaillierter Vergleich der Fitnesszuweisung und der Selektion multikriterieller GA wird in [Zit99] gegeben.

Der wesentliche Vorteil multikriterieller GA besteht in der Ermittlung der Pareto-Front in der gesamten Breite, ohne dass eine Skalarisierung des Zielfunktionsvektors und mehrere Optimierungsläufe notwendig sind. Somit kann der gesamte Lösungsraum untersucht und bei der Auswahl einer Lösung verschiedene Zielkriterien gegenübergestellt werden. Dies führt jedoch zu einer großen Anzahl an Evaluationen [MA04]. Zudem kann nach einer Optimierung nicht eindeutig festgestellt werden, ob die gefundenen Lösungen wirklich die exakte Pareto-Front darstellen. Die Lösungen repräsentieren lediglich die Individuen, zu denen im Verlauf der Optimierung keine dominierenden Individuen gefunden wurden. Somit kann für ein Individuum der Pareto-Front keine eindeutige Aussage getroffen werden, ob dieses ein lokales oder globales Optimum darstellt. Dies kann nur mit einer gewissen Wahrscheinlichkeit angenommen werden [Har14].

2.8.1.5 Abschließende Bemerkungen

Sowohl bei der Skalarisierung des Zielfunktionsvektors und der Durchführung mehrerer Optimierungsläufe oder der Verwendung eines multikriteriellen GA werden mehr Evaluationen benötigt als bei einer monokriteriellen Optimierung [MA04]. Durch die in dieser

Arbeit vorgestellten Parallelisierungsmethoden kann jedoch die Effizienz multikriterieller Optimierungen gesteigert werden (s. Kapitel 4).

Bei der Ermittlung der Menge Pareto-optimaler Lösungen durch multikriterielle Optimierungsstrategien wird keine konkrete Lösung für eine Optimierungsaufgabe erzielt, sondern die Anzahl der zu untersuchenden Varianten gemäß der Zielkriterien auf die Pareto-optimalen Lösungen reduziert. Die Auswahl einer oder mehrerer geeigneter Lösungen erfolgt durch den Anwender [KMKM09]. Lösungen multikriterieller Optimierungsaufgaben stellen somit auch die Basis von Entscheidungsmodellen dar [Gel14].

2.8.2 Multidisziplinäre Optimierung

Multidisziplinäre Optimierungen sind durch die Einbindung verschiedener Disziplinen, Fachbereiche oder Subsysteme gekennzeichnet. Dabei gilt es, die Anforderungen und Zielkriterien der einzelnen Disziplinen sowie deren Interaktionen zu berücksichtigen. Ausgehend von Anwendungen in der Luftfahrtindustrie, in denen sowohl mechanische als auch strömungstechnische Anforderungen berücksichtigt werden mussten, ist die Bedeutung multidisziplinärer Optimierungsstrategien aufgrund zunehmender Produktkomplexität und Systemintegration stetig angestiegen [ML13]. Neben der Luftfahrtindustrie sind die meisten Anwendungsfälle in der Automobilindustrie zu finden [RBN15].

Aufgrund der Verwendung unterschiedlicher Disziplinen und Subsysteme kommen innerhalb des Evaluationsmodells verschiedene Modelle zum Einsatz, welche die Evaluationen und Lastfälle der involvierten Disziplinen abbilden. Die Designvariablen sind dabei auf die einzelnen Modelle verteilt und können modellspezifisch oder in mehreren Modellen verwendet werden. Die aus den Evaluationen der Modelle ermittelten Zustandsvariablen stellen Zielkriterien bzw. Restriktionen der Optimierung dar oder bilden Eingangsvariablen weiterer Modelle. Diese Variablen werden als Kopplungsvariablen bezeichnet [ML13]. Aus der Abhängigkeit der Modelle resultiert der mögliche Parallelisierungsgrad der Evaluationsprozesse. Existieren keine Kopplungsvariablen, hängen die Modelle und die damit verbundenen Evaluationsprozesse nur von den Designvariablen ab und können vollständig parallel bearbeitet werden. Existieren Abhängigkeiten der Variablen, müssen diese bei der Parallelisierung berücksichtigt werden.

Die aus den Evaluationen der einzelnen Modelle ermittelten Zustandsvariablen stellen meist auch die Zielkriterien der involvierten Disziplinen dar. Multidisziplinäre Optimierungsaufgaben besitzen somit oft auch multikriterielle Zielstellungen. Aber auch die Verwendung der Zustandsvariablen als Restriktionen in einer monokriteriellen Optimierung ist möglich, z. B. bei der Minimierung der Masse [RBN15].

Da bei der multidisziplinären Optimierung primär das Evaluationsmodell zu berücksichtigen ist, kann die Wahl des Optimierungsverfahrens weitestgehend unabhängig von der Multidisziplinarität, jedoch ggf. unter Berücksichtigung multikriterieller Optimierungsstrategien erfolgen (s. Abschnitt 2.8.2). Ausgehend von verschiedenen Problemstellungen und Anwendungsfällen wurden unterschiedliche Strategien zur multidisziplinären Optimierung entwickelt. Aufgrund der Beteiligung unterschiedlicher Fachbereiche werden dabei neben dem eigentlichen Optimierungsproblem auch die Organisationsstruktur sowie die Kommunikation und der Datenaustausch zwischen den Fachbereichen berücksichtigt. Multidisziplinäre Optimierungsstrategien werden primär anhand der in der Optimierung

implementierten Stufen klassifiziert. Dabei werden einstufige und mehrstufige Optimierungsstrategien unterschieden [ML13], [RBN15]. Die wesentlichen Strategien werden im Folgenden vorgestellt. Eine vergleichende Betrachtung aller multidisziplinärer Optimierungsstrategien ist in [ML13] gegeben.

Einstufige Optimierungsstrategien (Single-Level): Bei diesen Optimierungsstrategien wird ein zentrales Optimierungssystem verwendet, welches die Designvariablen generiert und die Zielkriterien auswertet [RBN15]. Die Optimierungen werden daher auch als *monolithisch* bezeichnet [ML13]. Das verwendete Optimierungssystem muss die Möglichkeit bieten, alle benötigten Modelle des Evaluationsmodells einzubinden. Da bei dieser Art von Optimierungssystemen das Evaluations- und das Optimierungsmodell nicht gekoppelt sind, sondern unabhängig voneinander agieren können, werden diese Systeme als externe Optimierungssysteme bzw. *General Purpose*-Optimierer bezeichnet [Har14]. Beispielhafte Anwendungen einstufiger Optimierungsstrategien werden in [NL05], [Sch13], [Har14] beschreiben.

Ein wesentlicher Vorteil einstufiger Optimierungsstrategien ist die einfache Anwendung. Zudem können vor allem in frühen Entwicklungsphasen schnell multidisziplinäre Zielstellungen gegenübergestellt werden. Jedoch erfordern diese Strategien insbesondere bei großen Organisationen und einer großen Anzahl involvierter Disziplinen und Lastfälle einen hohen Aufwand an Kommunikation und Vorbereitung, da die benötigten Designvariablen, Evaluationsmodelle, Zielkriterien und Restriktionen zentral erfasst werden müssen. Zudem resultiert eine große Anzahl an Disziplinen und Lastfällen auch in einer großen Anzahl an Designvariablen, da meist jede Disziplin unterschiedliche Designvariablen verwendet. Dies führt oft zu einer großen Anzahl an Evaluationen. Da das verwendete Optimierungsverfahren bezogen auf die gesamte Zielstellung der Optimierung ausgewählt wird, stellt dies bezogen auf die Einzeldisziplinen oft einen Kompromiss dar. Bei einer überschaubaren Anzahl involvierter Disziplinen und Lastfälle sind einstufige Optimierungsstrategien mehrstufigen aufgrund der einfachen Anwendung vorzuziehen.

Mehrstufige Optimierungsstrategien (Multi-Level): Diese Optimierungsstrategien sind durch eine Zerlegung und Verteilung des Optimierungsprozesses charakterisiert. Dabei werden meist zwei Stufen verwendet, aber auch die Verwendung mehrerer Stufen ist möglich. Die Optimierung bezogen auf die Zielkriterien des Gesamtsystems erfolgt auf der oberen Stufe, der Systemebene. Die Optimierungen der Subsysteme, Komponenten und Disziplinen erfolgt in der darunterliegenden Stufe [RBN15]. Je nach eingesetzter Optimierungsstrategie und Zielstellung, werden während der Optimierung zwischen den Stufen verschiedene Variablen ausgetauscht (Kopplungsvariablen).

Folgende multidisziplinäre Optimierungsstrategien haben sich insbesondere in der Luftfahrtindustrie etabliert:

- Concurrent Subspace Optimization [SS89], [BHSS92], [WRBB96]
- Collaborative Optimization [Bra96], [BK97], [SK00]
- Bilevel Integrated System Synthesis [SSAS00], [SSAPS03]
- Analytical Target Cascading [Kim01], [KMPJ03]

Mehrstufige Optimierungsstrategien sind vor allem bei großen Organisationen und einer Vielzahl an beteiligten Disziplinen sinnvoll, da die notwendige Kommunikation und

der Datenaustausch zwischen den beteiligten Bereichen reduziert werden. Die jeweiligen Disziplinen agieren unabhängig voneinander. Somit können je nach Disziplin eigene Designvariablen, Evaluationsmodelle, Zielkriterien und Restriktionen verwendet werden. Zudem können disziplinspezifische Optimierungsverfahren eingesetzt werden. Nachteilig ist die Komplexität dieser Strategien und der damit verbundene Aufwand bei der Entwicklung der Systemebene und zum Austausch der relevanten Variablen. Zudem besteht aufgrund der Komplexität der Verfahren die Gefahr, das globale Optimum des Gesamtsystems zu übergehen. Weiterhin werden durch die verschiedenen Optimierungen in der zweiten Stufe insgesamt oft mehr Evaluationen und somit mehr Ressourcen für die Bearbeitung der Evaluationsprozesse benötigt als bei einstufigen Strategien [ML13].

Durch die Verwendung von Metamodellen (s. Abschnitt 2.7) können die Vorteile beider multidisziplinärer Optimierungsstrategien genutzt werden. Die Generierung der Metamodelle erfolgt dabei fachdisziplin- und lastfallspezifisch in den jeweiligen Abteilungen. Die einzelnen Metamodelle werden anschließend als Evaluationsmodelle in einer einstufigen Optimierung des Gesamtsystems verwendet. Die Designvariablen und Zielkriterien müssen zentral erfasst werden. Insbesondere bei komplexen multidisziplinären Optimierungen kann somit der numerische Aufwand deutlich reduziert werden [SK00], [RBN15]. Auch bei den Fallstudien im Rahmen der vorliegenden Arbeit wird eine einstufige Optimierungsstrategie verfolgt (s. Abschnitt 5.5).

2.8.3 Kombination von Optimierungsmethoden und -verfahren

Durch die Kombination von Optimierungsmethoden und -verfahren können die spezifischen Vorteile der jeweiligen Methoden und Verfahren genutzt werden. Somit lassen sich die Effektivität oder auch die Effizienz der Methoden und Verfahren steigern. Die Effektivität beschreibt hierbei die Qualität der erzielten Lösung. Die Effizienz gibt an, wie schnell diese Lösung erreicht wird. Dabei werden die Anzahl der benötigten Evaluationen und der mögliche Parallelisierungsgrad bezogen auf die zur Verfügung stehenden Ressourcen berücksichtigt (s. Abschnitt 2.1). Kombinationen von Optimierungsmethoden und -verfahren können in den folgenden Klassen unterschieden werden:

Sequentielle Kombination: Dies stellt die einfachste Form der Kombination dar. Dabei wird zunächst mit einer Methode bzw. einem Verfahren begonnen und nach einer gewissen Zeit, Anzahl von Iterationen oder einem anderen Kriterium in eine andere Methode oder Verfahren gewechselt. So kann z. B. mit einer Topologieoptimierung begonnen werden, deren Ergebnisse anschließend in einer Formoptimierung weiterverarbeitet werden [RMS98]. Der Wechsel kann automatisch oder manuell durch den Anwender erfolgen. Durch diese Kombination lässt sich primär die Effektivität steigern, da durch die verschiedenen Methoden unterschiedliche Disziplinen des zu optimierenden Produktes angesprochen werden und somit im Vergleich zur Verwendung einer einzelnen Methode ein größerer Lösungsraum berücksichtigt wird.

Eine weitere Form der sequentiellen Kombination sind hybride Optimierungsverfahren (s. Abschnitt 2.6.3). Durch diese kann zum einen die Effektivität gesteigert werden, da z. B. durch die Verwendung eines Gradientenverfahren im Anschluss an ein stochastisches Verfahren eine höhere Wahrscheinlichkeit besteht, das globale Optimum des Problems zu

finden. Zum anderen lässt sich dadurch auch die Effizienz steigern, da Gradientenverfahren deutlich weniger Evaluationen benötigen als stochastische Verfahren.

Auch bei der Optimierung basierend auf Metamodellen werden meist deutlich weniger Evaluationen benötigt, als z. B. bei rein stochastischen Optimierungsverfahren. Daher kommen Metamodelle insbesondere bei numerisch sehr aufwendigen Evaluationen zum Einsatz (s. Abschnitt 2.7).

Parallele Kombination: Diese Form kommt primär bei der Kombination von Optimierungsverfahren zum Einsatz. Parallele Kombinationen von Optimierungsmethoden sind eher unüblich, da sich die Definition der Designvariablen und die durch die Optimierung angesprochenen Disziplinen deutlich unterscheiden.

Durch die parallele Verwendung verschiedener Optimierungsalgorithmen kann die Wahrscheinlichkeit erhöht werden, das globale Optimum eines Problems zu finden, insbesondere bei der erstmaligen Optimierung des Problems. Dabei wird ein globales Abbruchkriterium verwendet, in das die jeweiligen Lösungen der einzelnen Algorithmen einfließen [GH02]. Somit lässt sich die Effektivität im Vergleich zu einem einzelnen Algorithmus erhöhen, jedoch zulasten der Effizienz. Diese Kombinationsform ist sehr rechenintensiv und mit einer hohen Anzahl verfügbarer Ressourcen verbunden. Da die Evaluationen der einzelnen Algorithmen unabhängig voneinander sind, können diese auch parallelisiert werden.

Die Effizienz einer Optimierung kann durch die parallele Verwendung von Metamodellen erhöht werden, insbesondere bei einer multidisziplinären Zielstellung und einer variablen Anzahl zur Verfügung stehender Ressourcen. Die bei der Optimierung durchgeführten Evaluationen dienen dabei als Stützstellen zur parallelen Generierung eines Metamodells. Ist zu einem bestimmten Zeitpunkt keine Evaluation auf Basis des Evaluationsmodells möglich, z. B. beim Ausfall einer Ressource, erfolgt die Evaluation durch Verwendung des Metamodells. Somit können bei intensiv genutzten Ressourcen Wartezeiten vermieden und eine geringe Gesamtlaufzeit der Optimierung gewährleistet werden [RB13].

Auch die parallele Durchführung mehrerer Optimierungsläufe zur Bestimmung der Pareto-Front einer multikriteriellen Optimierung stellt eine parallele Kombination dar. Obwohl die Anzahl der Evaluationen dadurch nicht reduziert wird, kann bei einer ausreichenden Anzahl an Ressourcen, die Effizienz der Optimierung durch die parallele Bearbeitung erhöht werden.

Integrierte Kombination: Bei diesen Kombinationen werden die Optimierungsmethoden und -verfahren ineinander integriert bzw. verschachtelt durchgeführt. Daher werden diese auch als Multi-Level-Optimierungen bezeichnet [Sch13].

Durch die integrierte Kombination von Optimierungsverfahren lassen sich Systemoptimierungen und multidisziplinäre Zielstellungen realisieren. Somit kann die Effizienz der Verfahren erhöht werden, da bei sehr großen Systemen und einer Vielzahl beteiligter Fachdisziplinen der Kommunikationsaufwand reduziert wird und fachdisziplinspezifische Optimierungsverfahren eingesetzt werden können (s. Abschnitt 2.8.2).

Integrierte Kombinationen von Optimierungsmethoden dienen der Erhöhung der Effektivität. Analog zu sequentiellen Kombinationen werden durch die verschiedenen Methoden unterschiedliche Disziplinen des zu optimierenden Produktes angesprochen, wodurch im

Vergleich zur Verwendung einer einzelnen Methode ein größerer Lösungsraum berücksichtigt wird. Da die Kombination integriert erfolgt, können auch Wechselwirkungen zwischen den Optimierungsmethoden berücksichtigt werden. Charakterisiert sind diese Kombinationen durch einen äußeren und einen oder mehrere innere iterative Optimierungsprozesse. So können simultan die Form und die Topologie eines Bauteils optimiert werden. Die Definition der Designvariablen erfolgt dabei implizit durch das System [ACTR02], [ACTR04].

Auch die Integration einer Topologieoptimierung in eine Parameteroptimierung ist durch diese Kombination möglich. Durch die Parameteroptimierung als äußeren und die Topologieoptimierung als inneren Prozess kann z. B. während der Parameteroptimierung von Getriebegehäusen für jede Variante die durch die Topologieoptimierung ermittelte optimale Rippengeometrie berücksichtigt werden. Da die Rippengeometrie einen großen Einfluss auf die Gehäusesteifigkeit hat und wiederum eine Verringerung der Wandstärke ermöglicht, führt die integrierte Kombination der Methoden zu einer besseren Lösung als die rein sequentielle Kombination [WTVS13].

Aufgrund des integrierten Ablaufs der Optimierungsprozesse erfordern diese Kombinationen oft deutlich mehr Iterationen als sequentielle Kombinationen. So ergibt sich die Anzahl der Iterationen sequentieller Kombinationen aus der *Addition* der Iterationen der einzelnen Optimierungsmethoden. Die Iterationsanzahl integrierter Kombinationen hingegen resultiert aus der *Multiplikation* der Iterationen der kombinierten Optimierungsmethoden.

Die Kombination von Optimierungsmethoden und -verfahren bietet Potential zur Erhöhung der Effektivität oder der Effizienz im Vergleich zur Anwendung einzelner Methoden und Verfahren. Kommen dabei Optimierungsverfahren zum Einsatz, welche in der einzelnen Anwendung parallel bearbeitet werden können (z. B. stochastische Optimierungsverfahren), ist auch bei der Kombination die parallele Bearbeitung dieser Verfahren möglich, wodurch ebenfalls die Effizienz gesteigert werden kann. Im Rahmen der vorliegenden Arbeit wird daher bei der Formulierung der Anforderungen und bei der Umsetzung des Optimierungssystems zusätzlich die Möglichkeit der Kombination von Optimierungsmethoden und -verfahren berücksichtigt.

3 Analogiebetrachtungen von Optimierungsverfahren und Produktentwicklung

In diesem Kapitel werden Analogien beschrieben, welche die Basis für die Entwicklung von Optimierungsverfahren in der Produktentwicklung darstellen sowie die Grundlage für Vorgehensmodelle der Produktentwicklung bilden. Dabei wird zunächst die biologisch inspirierte Programmierung erläutert. Diese beinhaltet Verfahren und Algorithmen, welche durch Vorgänge in der Natur inspiriert oder direkt aus diesen abgeleitet wurden und somit gleichermaßen die Grundlage einer Vielzahl stochastischer Optimierungsverfahren repräsentieren. Besonderer Fokus liegt dabei auf Analogiebetrachtungen evolutionärer Algorithmen, da diese auch bei den im weiteren Verlauf dieser Arbeit entwickelten Parallelisierungsmethoden Anwendung finden.

Anschließend werden Optimierungsverfahren vorgestellt, welche durch Analogien zum Verhalten interdisziplinärer Entwicklungsteams bei der Produktentwicklung inspiriert sind. Dieser Ansatz wird im weiteren Verlauf der Arbeit aufgegriffen und weiterverfolgt. Im Gegensatz zu den in diesem Kapitel beschriebenen Verfahren, welche die Generierung von Lösungen eines bestimmten Problems zum Ziel haben, basiert die in dieser Arbeit entwickelte Parallelisierungsmethode auf Analogien zur effizienten und robusten Bearbeitung von Aufgaben und Tätigkeiten innerhalb interdisziplinärer Entwicklungsteams, um somit die Durchführung von Optimierungen effizienter zu gestalten (s. Kapitel 4).

Weiterhin erfolgt in diesem Kapitel eine Beschreibung der Autogenetischen Konstruktionstheorie. Ausgehend von einer umfassenden Analogiebetrachtung von Prozessen der Produktentwicklung und den Vorgängen der biologischen Evolution wird hierbei die gesamte Produktentwicklung als evolutionärer Prozess verstanden. Da die Produktentwicklung ebenso als iterativer Optimierungsprozess angesehen werden kann und die Vorgänge der biologischen Evolution die Inspiration verschiedener evolutionärer Algorithmen darstellen, beinhaltet die Anwendung der Autogenetischen Konstruktionstheorie verschiedene Optimierungsstrategien unter Nutzung evolutionärer und genetischer Algorithmen. Insbesondere bei komplexen Evaluationsprozessen ist somit auch in diesem Umfeld der Produktentwicklung eine Effizienzsteigerung durch die entwickelten Parallelisierungsmethoden möglich.

3.1 Biologisch inspirierte Programmierung

Die biologisch inspirierte Programmierung beinhaltet Verfahren und Algorithmen, welche durch Vorgänge in der Natur inspiriert oder direkt aus diesen abgeleitet wurden. Hierbei werden auch die Begriffe evolutionäre Programmierung und evolutionäre Berechnung verwendet [FOW66], [BFM97], [SD08]. Da diese Bezeichnungen primär den evolutionären Aspekt berücksichtigen, muss der Begriff weiter gefasst werden, um auch andere von der Natur inspirierte Verfahren zu berücksichtigen, z. B. den Ameisenkolonie-Algorithmus [CDM92]. Analogien zu Vorgängen in der Natur bilden die Basis einer Vielzahl weiterer stochastischer Optimierungsverfahren. Diese wurden bereits in Abschnitt 2.6.2 aufgeführt.

Evolutionäre Algorithmen (EA) stellen die wichtigste Gruppe stochastischer Optimierungsverfahren dar. Sie basieren auf Analogien zur biologischen Evolution und bilden eine Teilmenge der biologisch inspirierten Programmierung. Dabei dient die biologische Evolution eher als Orientierung und Inspiration. Eine detaillierte Nachbildung aller ablaufenden Vorgänge wird nicht verfolgt. Folgende Eigenschaften der biologischen Evolution liegen EA zugrunde [GKK04]:

- Anpassung an eine sich verändernde Umgebung durch Adaption
- Lösen von Problemen auf völlig neue Art und Weise durch Innovation
- Erzeugen von Lösungen komplexer Probleme in kurzer Zeit und durch Prüfung einer relativ geringen Anzahl von Lösungsmöglichkeiten

Diese Eigenschaften werden innerhalb der Algorithmen in abstrakter Form abgebildet. EA stellen somit ein sehr schematisches und abstraktes Modell der biologischen Evolution dar [GKK04]. Dabei finden sich die wesentlichen Begriffe und Objekte der biologischen Evolution auch bei den EA wieder. Diese werden in Tabelle 3.1 gegenübergestellt.

Tabelle 3.1: Analogien der biologischen Evolution und evolutionärer Algorithmen, nach [Gol89], [GKK04]

Bezeichnung	Biologische Evolution	Evolutionäre Algorithmen
Chromosom	Festlegung der Eigenschaften eines Organismus (bzw. eines Teils davon)	Zeichenkette (String), Liste
Gen	Teilstück eines Chromosoms, das eine Eigenschaft festlegt, z. B. Gen für die Augenfarbe	Zeichen, Variable, Feature
Allel	Ausprägung eines Gens, z. B. Augenfarbe blau	Wert eines Zeichens oder einer Variable, Ausprägung eines Features
Locus	Physische Position eines Gens im Chromosom	Position eines Zeichens in einer Zeichenkette
Genotyp	Erscheinungsbild des Chromosoms eines Organismus	Formale Codierung einer Variante/ Individuums
Phänotyp	Beobachtbares Erscheinungsbild eines Organismus	Decodierte Ausprägung einer Variante/Individuums, z. B. Designvariablenwerte, Modell
Population	Menge von Organismen	Menge von Chromosomen/ Individuen
Fitness	Überlebens-/Vermehrungsfähigkeit eines Organismus	Güte einer durch ein Chromosom codierten Variante/Individuum, Maß für die Selektionswahrscheinlichkeit
Generation	Population zu einem Zeitpunkt	Population zu einem Zeitpunkt
Reproduktion	Erzeugen von Nachkommen aus einem oder mehreren Organismen	Erzeugen von Chromosomen aus einem oder mehreren Chromosomen/Individuen

Bedingt durch die der biologischen Evolution nachempfundene Funktionsweise von EA werden auch die Eigenschaften des zu optimierenden Objektes, z. B. ein Produkt, durch Gene beschrieben. In der biologischen Evolution werden die Eigenschaften von Organismen durch Gene definiert, welche in Chromosomen zusammengefasst werden. Ein Gen bestimmt dabei eine Eigenschaft des Organismus. Die Position eines Gens im Chromosom wird als Locus bezeichnet. Bei EA kann ein Chromosom durch unterschiedliche Datentypen und -strukturen repräsentiert werden, z. B. durch Zeichenketten oder Listen. Je nach gewählter Repräsentation werden die Eigenschaften des zu optimierenden Objektes durch Zeichen, Variablen oder Features beschrieben. Die konkrete Ausprägung einer Eigenschaft wird als Allel bezeichnet. Dies kann z. B. der Wert einer Variable oder die Ausprägung eines Features sein.

Der Genotyp beschreibt die konkrete Ausprägung eines Chromosoms. Er bildet somit die Kombination aller Allele und Loci eines Organismus ab. In einem EA entspricht dies der formalen Codierung einer Variante des zu optimierenden Objektes, welche auch als Individuum bezeichnet wird. Die decodierte Form des Individuums wird als Phänotyp bezeichnet. Dieser kann z. B. in Form konkreter Designvariablenwerte oder als Modell vorliegen. Die Begriffe Population, Fitness, Generation und Reproduktion wurden bereits bei der Erläuterung der Funktionsweise von EA beschrieben und werden daher an dieser Stelle nicht gesondert aufgeführt (s. Abschnitt 2.6.2.1).

Neben Vorgängen aus der Natur, welche das Vorbild einer Vielzahl stochastischer Optimierungsverfahren darstellen, können auch Analogien zu Vorgängen in anderen Bereichen die Basis für die Entwicklung von Optimierungsverfahren darstellen. So kann auch das Verhalten von Teammitgliedern eines Entwicklungsteams als Inspiration für Optimierungsverfahren dienen. Die Teammitglieder werden dabei durch sog. Agenten abgebildet [CCK99]. Agenten sind Komponenten von Softwarearchitekturen, welche in einem definierten Bereich unabhängig und selbstständig agieren und Teilprobleme lösen können [FH11]. Da sich die einzelnen Agenten analog zu Spezialisten unterschiedlicher Fachrichtungen verhalten und auch unterschiedliche Ziele verfolgen können, lassen sich durch die Implementierung spezifischer Verhaltensregeln und Logiken auch wissensbasierte Aspekte realisieren [CCK99].

Beispielhaft hierzu ist der *Heterogeneous Simulated Annealing Team* (HSAT)-Algorithmus zu nennen. Dieser basiert auf den folgenden wesentlichen Eigenschaften von Entwicklungsteams bei der Lösungsfindung [MCK15a]:

- Selbstständige, asynchrone Lösungssuche durch einzelne Teammitglieder
- Interaktion zwischen Teammitgliedern zur Auswahl und Verbesserung von Lösungen

Ausgehend von einer Problemstellung werden durch ein Entwicklungsteam in der Regel zunächst viele verschiedene mögliche Lösungen generiert (Divergenz), welche anschließend bezogen auf die Problemstellung zusammengeführt und fokussiert werden (Konvergenz). Die Lösungen konvergieren somit in einer begrenzten Anzahl bzw. Menge [Ma07], [FCK10]. Auch bei sehr präzise und strikt formulierten Problemstellungen entstehen dabei oft verschiedene Lösungen, welche sich in einigen Aspekten unterscheiden [MCK15b]. Zusätzlich zur individuellen Problemlösung durch die Teammitglieder, bei der jedes Teammitglied nach einer eigenen Strategie handelt, führen Interaktion und Kommunikation zwischen den Teammitgliedern zu weiteren Verbesserungen der Lösungen [WCF⁺14].

Dieses Vorgehen wird durch den HSAT-Algorithmus abgebildet. Die einzelnen Mitglieder des Entwicklungsteams werden durch Agenten repräsentiert. Jeder Agent führt eine eigene Optimierung unter Verwendung des Simulated Annealing-Verfahrens durch (s. Abschnitt 2.6.2.3). Die Agenten unterscheiden sich im zeitlichen Verlauf und in der Anpassungsfähigkeit der Abkühlvorgänge. Somit wird die individuelle und heterogene Problemlösung realisiert. Die Interaktion zwischen den Agenten erfolgt in Abhängigkeit von der Qualität der Lösungen. Je stärker sich die Lösungen der Agenten bezogen auf den Zielfunktionswert unterscheiden, desto höher ist die Wahrscheinlichkeit, dass diese ausgetauscht oder schlechte Lösungen überschrieben werden. Bei kleineren Unterschieden der Lösungen ist die Wahrscheinlichkeit des Austauschs geringer. Somit besteht die Möglichkeit, dass die Agenten unterschiedliche Lösungen generieren, welche sich in ihrer Qualität nur geringfügig unterscheiden und somit nahezu gleichwertig sind [MCK15a].

Da die Agenten unabhängig voneinander arbeiten und Interaktionen in unregelmäßigen Abständen erfolgen, können die Optimierungen der Agenten parallel durchgeführt werden. Eine weitere Effizienzsteigerung dieses Optimierungsverfahrens kann, insbesondere bei komplexen Evaluationsprozessen, durch die in dieser Arbeit vorgestellten Parallelisierungsmethoden erzielt werden (s. Kapitel 4).

Eine weitere Möglichkeit der Entwicklung von Optimierungsverfahren auf Basis von Analogien zum menschlichen Problemlösen ist die Analyse und Nachbildung des Vorgehens bei der manuellen Lösung einer Optimierungsaufgabe durch einen Produktentwickler. Die Beschreibung des Vorgehens zur Problemlösung erfolgt dabei regelbasiert auf Basis einer Analyse erfolgreicher und nicht erfolgreicher Vorgehensweisen. Durch die Implementierung von drei erfolgreichen Vorgehensweisen in unabhängig im Hintergrund agierende Software-Agenten, in denen die Vorgehensweisen iterativ auf die aktuelle Problemstellung angepasst werden, wird der Produktentwickler bei der Lösungsfindung gleichartiger Problemstellungen unterstützt, was zur Ermittlung besserer Lösungen führt [ECSL14], [ECSL15].

Dies zeigt, dass auch Analogien zu Produktentwicklern und Entwicklungsteams die Basis für effektive und effiziente Optimierungsverfahren bilden können, da die Aktivitäten und Prozesse der Produktentwicklung ebenfalls als iterativer Optimierungsprozess basierend auf dem TOTE-Schema (s. Abschnitt 2.3) beschrieben werden können. Dabei werden durch die Verfahren insbesondere die relevanten Prozesse zum Lösen einer Optimierungsaufgabe abgebildet, welche auch bei einer manuellen Optimierung durchgeführt werden.

Diese Analogiebetrachtung stellt die Basis der in Abschnitt 4.3 entwickelten Parallelisierungsmethoden Concurrent und Simultaneous Optimization dar. Bei der Durchführung einer multidisziplinären Optimierung sollen sich die Einheiten der verwendeten verteilten IT-Umgebung analog zu den Mitgliedern eines interdisziplinären Teams verhalten, welche unterschiedliche Qualifikationen besitzen und je nach Qualifikation unterschiedliche Aufgaben bearbeiten können. Das Ziel der Organisation und Aufgabenverteilung ist dabei, die gemeinsame Gesamtaufgabe möglichst effizient zu bearbeiten und Leerlauf in den Einheiten zu minimieren. Zudem sollen selbstständig und dynamisch Ausfälle von Einheiten kompensiert und zusätzliche Einheiten integriert werden können.

3.2 Autogenetische Konstruktionstheorie

Eine umfassendere Analogiebetrachtung zur biologischen Evolution liefert die Autogenetische Konstruktionstheorie (AKT). Dabei wird sich nicht nur auf Optimierungsverfahren beschränkt, sondern die gesamte Produktentwicklung als evolutionärer Prozess betrachtet. Im Gegensatz zur Bionik, bei der die Ergebnisse der biologischen Evolution auf technische Produkte angewendet werden, z. B. die Struktur von Blättern, werden bei der AKT die Vorgänge der Evolution auf die Produktentwicklung mit dem Ziel übertragen, eine umfassende und ganzheitliche Beschreibung der Produktentwicklung sowie der involvierten Prozesse, Anforderungen, Randbedingungen und Objekte (einschließlich ihrer Eigenschaften) zu erzeugen [BV94a], [BV94b], [VCJB05], [KFV11].

Diesem Ansatz liegen Analogien zwischen der biologischen Evolution und der Produktentwicklung zugrunde. Kennzeichnend für die biologische Evolution sind die kontinuierliche Weiterentwicklung und Anpassung von Organismen bzw. Individuen in einer sich verändernden Umgebung unter minimalem Einsatz von Ressourcen. Diese Eigenschaft kann auch durch das Prinzip von Versuch und Irrtum (engl. trial and error) beschrieben werden. Die Evolution stellt somit einen kontinuierlichen Verbesserungsprozess und eine Art iterative Optimierung gemäß des TOTE-Schemas dar (s. Abschnitt 2.3), was auch auf die Produktentwicklung zutrifft. Hierbei unterliegen die geltenden Anforderungen, Randbedingungen und Restriktionen dynamischen Änderungen und können sich zudem widersprechen [VCJB05].

Die Erzeugung von Individuen wird in der biologischen Evolution durch die evolutionären Operatoren Selektion, Replikation, Rekombination und Mutation beschrieben. Diese finden auch in evolutionären Algorithmen Anwendung (s. Abschnitt 2.6.2.1) und werden in der AKT zur Unterstützung der Tätigkeiten der Produktentwicklung genutzt. Gemäß der unterschiedlichen Tätigkeiten können dabei verschiedene Operatoren zum Einsatz kommen. Diese sind in Tabelle 3.2 dargestellt.

Tabelle 3.2: Unterstützung von Tätigkeiten der Produktentwicklung durch evolutionäre Operatoren, nach [VCJB05], [VWBZ09]

Tätigkeit in der Produktentwicklung	Evolutionäre Operatoren
Finden und Verwenden geeigneter Lösungen	Selektion, Replikation
Neuer Gedanke, Ideengenerierung	(Zufall,) Mutation, Rekombination
Bewerten einer Lösung	Selektion
Verändern einer Lösung	Mutation, Rekombination
Festhalten von Teillösungen	Selektion, Replikation
Detaillieren von Lösungen	Selektion, Replikation, Rekombination

Die AKT beinhaltet eine produkt- und prozessbezogene Sichtweise. Eine Gegenüberstellung produkt- und prozessbezogener Aspekte der Evolution und der Produktentwicklung wird in [VJK13] gegeben. Dabei werden auch die Schwierigkeiten bei der Erstellung eines allgemeingültigen Analogiemodells erläutert und am Beispiel von Material- und Struktureigenschaften prinzipielle Unterschiede beider Bereiche beschrieben.

Wie eingangs beschrieben, wird die AKT als ganzheitlicher Ansatz der Produktentwicklung nach dem Vorbild der biologischen Evolution verstanden. Da diese als iterativer Optimierungsprozess angesehen werden kann und die Vorgänge der biologischen Evolution auch die Inspiration evolutionärer Algorithmen darstellen (s. Abschnitt 3.1), beinhaltet die AKT verschiedene Optimierungsstrategien unter Anwendung evolutionärer und genetischer Algorithmen. Eine Art der Implementierung der AKT in den Produktentwicklungsprozess stellt das Optimierungssystem NOA dar, welches verschiedene Module evolutionärer und genetischer Algorithmen beinhaltet [JC04]. Anwendungen verschiedener Detaillierungs- und Komplexitätsgrade finden sich in [VCJB05], [VENKJ07], [KfV11].

Da NOA die wesentlichen Eigenschaften zur Parallelisierung evolutionärer und genetischer Algorithmen durch Modelldekomposition besitzt (s. Abschnitt 4.1), kann durch Parallelisierung die Effizienz einer Optimierung gesteigert werden. Insbesondere bei komplexen Evaluationsprozessen ist durch die in dieser Arbeit entwickelte Parallelisierungsmethode eine Effizienzsteigerung möglich (s. Kapitel 4). Auch bei dem in der prototypischen Umsetzung entwickelten Framework zur Realisierung der Parallelisierungsmethode wird NOA als Optimierungssystem eingesetzt (s. Kapitel 5).

In den folgenden Abschnitten wird der aktuelle Forschungsstand der AKT beschrieben. Dies beinhaltet das Produktmodell, den Lösungsraum und die Anwendung der AKT in den frühen Phasen der Produktentwicklung durch interaktive Evaluation. Das prinzipielle Vorgehensmodell der AKT wird ausführlich in [VK10], [VKB11], [VJK13] beschrieben und daher nicht gesondert aufgeführt.

Bei der Beschreibung des Produktmodells wird die Struktur des Produktmodells leicht verändert und somit eine bessere Abgrenzung zwischen den Produktparametern hergestellt, wodurch die Konsistenz innerhalb des Produktmodells erhöht wird. Zudem werden Ansätze zur praktischen Umsetzung des Produktmodells erarbeitet, um somit die Anwendung im Produktentwicklungsprozess zu unterstützen. Dabei wird auch die Verwendung der in dieser Arbeit entwickelten Parallelisierungsmethode herausgestellt.

Die Produktentwicklung sollte stets in einem interdisziplinären Team stattfinden [Vaj14], was auch für die Anwendung der AKT Gültigkeit hat. In den folgenden Abschnitten wird der Begriff des Produktentwicklers daher auch als Synonym für ein interdisziplinäres Entwicklungsteam verwendet.

3.2.1 Produktmodell

Das Ziel bei der Entwicklung des Produktmodells der AKT besteht darin, alle relevanten Daten eines Produktes zu erfassen und das Produkt sowie dessen Lebenszyklus vollständig zu beschreiben. Im Gegensatz zu konventionellen Modellen, bei denen in den verschiedenen Phasen des Produktentwicklungsprozesses unterschiedliche Datenstrukturen zum Einsatz kommen, soll das Produktmodell der AKT über den gesamten Produktlebenszyklus verfügbar sein, sodass die relevanten Daten in den jeweiligen Phasen schrittweise hinzugefügt und angepasst werden können. Dabei sollen die Anforderungen sowie die Multidisziplinarität des Entwicklungsprozesses berücksichtigt und somit auch die Entwicklung mechatronischer Produkte unterstützt werden. Da sich durch die Integration verschiedener Fachdisziplinen die Anzahl der miteinander kombinierbaren Lösungen und somit die Anzahl der möglichen Gesamtlösungen stark erhöht, erfordert die Entwicklung des Produktmodells eine strukturierte Architektur, welche flexibel erweiterbar ist [KHVZ11].

Das aus dieser Zielstellung entwickelte Produktmodell basiert auf Produktparametern (PP), welche über Partialmodelle die Attribute eines Produktes abbilden. Diese Zuordnung erfolgt strukturiert in Form einer Matrix. Dabei sind die Attribute horizontal und die Partialmodelle vertikal angeordnet. Die Evolution des Produktmodells während der Produktentwicklung wird in einer dritten Dimension beschrieben. Eine schematische Darstellung des Produktmodells ist in Abbildung 3.1 zu sehen. Das Ziel dieser Darstellung ist es, die Elemente des Lösungsraumes zu strukturieren und somit die Übersichtlichkeit sowie die Handhabbarkeit der Elemente zu verbessern [VJK13].

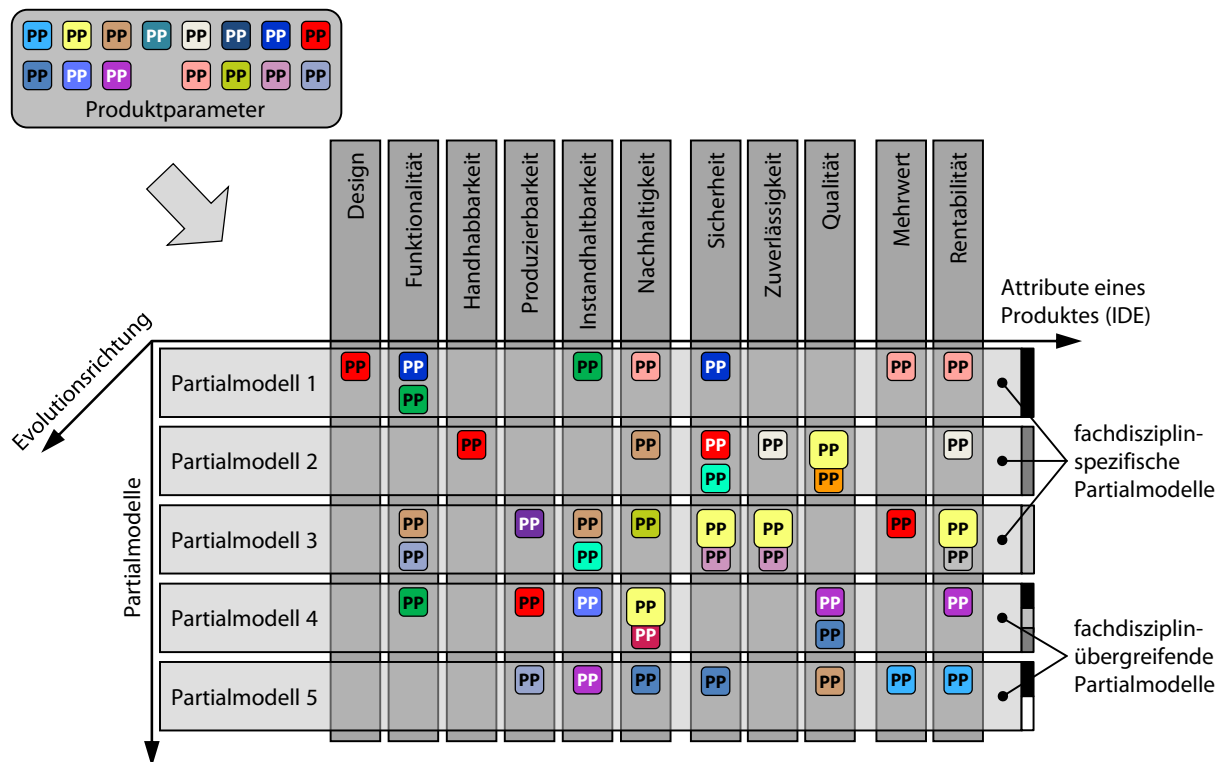


Abbildung 3.1: Produktmodell der AKT, nach [VJK13]

Die *Produktparameter* bilden die zentralen Elemente des Produktmodells. Sie beschreiben das gesamte Produkt und beeinflussen die Ausprägung der Produktattribute. Dabei repräsentieren die Produktparameter nicht nur geometrische Elemente des Produktes, sondern bilden auch eine Vielzahl weiterer Daten und Informationen ab, z. B. Fertigungsinformationen oder Kosten. Die Produktparameter werden in Anlehnung an die Klassifizierung nach Weber [Web05] gemäß ihrer Funktion in Merkmalen und Eigenschaften unterschieden. Merkmale können vom Produktentwickler direkt beeinflusst werden und charakterisieren die Struktur, die Gestalt und die Beschaffenheit eines Produktes. Eigenschaften hingegen definieren das Verhalten des Produktes und können vom Produktentwickler nicht direkt, sondern nur indirekt durch Veränderung der Merkmale beeinflusst werden [Web05]. Bei der Unterscheidung der Produktparameter werden die Merkmale des Produktes auch als Designparameter bezeichnet, da diese das Design, also die Konstruktion beeinflussen. Die Eigenschaften werden auch als Bewertungskriterien bezeichnet, da diese zur Bewertung eines Sachverhaltes bzw. eines Attributes herangezogen werden. Die Eigenschaften des Produktes bestimmen somit maßgeblich die Ausprägung der Produktattribute [VJK13].

Durch die Unterscheidung von Merkmalen und Eigenschaften werden analog zu den Designvariablen und Zustandsvariablen einer Optimierung (s. Abschnitt 2.1) die definierenden und die beschreibenden Eigenschaften des Produktes abgebildet.

Die *Partialmodelle* dienen der Ermittlung der Eigenschaften des Produktes aus den vom Produktentwickler festgelegten Merkmalen. Ein Partialmodell kann beliebig viele Merkmale als Eingangsparameter besitzen und beliebig viele Eigenschaften als Ergebnisparameter generieren. Ein Merkmal kann ebenso in beliebig vielen Partialmodellen verwendet werden. Eine Eigenschaft wird jedoch stets nur aus einem Partialmodell erzeugt. Diese Definition garantiert die spätere Nachverfolgung der Wirkzusammenhänge. Die Verwendung verschiedener Partialmodelle bedingt auch die Differenzierung der Produktparameter. Je nach Aufgabenstellung, dem zur Lösung verwendeten Partialmodell und dem aktuellen Zeitpunkt in der Produktentwicklung kann ein Produktparameter definierenden Charakter (Eingangsparameter, Merkmal) oder beschreibenden Charakter (Ergebnisparameter, Eigenschaft) besitzen. Dabei ist auch die zeitliche Veränderung der Parametercharakteristik im Verlauf der Produktentwicklung möglich.

Durch die Verwendung von Partialmodellen wird die komplexe Gesamtaufgabe der Entwicklung eines Produktes in weniger komplexe und somit beherrschbare Teilprobleme zerlegt. Die Zerlegung (Dekomposition) und Auswahl der benötigten Partialmodelle erfolgt durch den Produktentwickler. Dabei werden in einem Partialmodell die Produktparameter zusammengefasst, die für die Beschreibung und zur Lösung eines bestimmten Teilproblems relevant sind. Analog zu einer multidisziplinären Optimierungsstrategie, bei der verschiedene Evaluationsmodelle zum Einsatz kommen (s. Abschnitt 2.8.2), besteht auch das Produktmodell der AKT aus verschiedenen Partialmodellen. Beinhaltet die Beschreibung des Produktes verschiedene fachdisziplinspezifische Eigenschaften, z. B. bei mechatronischen Produkten, decken auch die Partialmodelle unterschiedliche Fachdisziplinen ab. Dabei wird zwischen fachdisziplinspezifischen und fachdisziplinübergreifenden Partialmodellen unterschieden [VJK13].

Fachdisziplinspezifische Partialmodelle repräsentieren nur eine jeweilige Fachdisziplin. Sie dienen daher der Bestimmung einer bzw. weniger Eigenschaften des Produktes. Fachdisziplinübergreifende Partialmodelle eignen sich zur Simulation eines Systemverhaltens und berücksichtigen Effekte und Zusammenhänge mehrerer Fachdisziplinen. Diese Modelle beinhalten eine größere Anzahl an Merkmalen und Eigenschaften. Folgende Modelle stehen beispielhaft für die verschiedenen Modellarten [VJK13]:

- Fachdisziplinspezifische Partialmodelle:
 - Berechnungsmodelle für die Finite Element Methode (Maschinenbau)
 - Modelle für den logischen Entwurf von Schaltkreisen (Elektrotechnik)
 - Modelle zur Beschreibung der Softwarearchitektur (Informatik)
- Fachdisziplinübergreifende Partialmodelle:
 - Funktionale Modelle zur Simulation von Systemeigenschaften (z. B. Mechatronics Concept Designer im CAx-System NX [Sie15])
 - Modelle zur Konzeptfindung durch interaktive Evaluation (s. Abschnitt 3.2.3)
 - Kopplungsmodelle verschiedener fachdisziplinspezifischer Modelle

Zur Bearbeitung bestimmter Teilprobleme werden vom Produktentwickler nur die für das Problem relevanten Partialmodelle benötigt. Die Modellauswahl bedingt die Auswahl der problemrelevanten Produktparameter. In Abhängigkeit von den involvierten Partialmodellen können auch die Produktparameter fachdisziplinspezifisch oder fachdisziplinübergreifend sein [VJK13]. Wie in Abbildung 3.1 beispielhaft für den in der Größe hervorgehobenen gelben Produktparameter dargestellt ist, können die Produktparameter auch in beiden Modellarten verwendet werden.

Die *Attribute* dienen der Beschreibung des Verhaltens des zu entwickelnden Produktes. Dieser aus dem Forschungsgebiet des Integrated Design Engineerings (IDE) übernommene Ansatz beinhaltet die vollständige Beschreibung und Bewertung eines Produktes hinsichtlich der an das Produkt gestellten Anforderungen. Die Attribute repräsentieren dabei verschiedene Sichtweisen auf das Produkt, welche gemäß der Anforderungen in unterschiedlichen Ausprägungen vorliegen müssen, um die Anforderungen zu erfüllen oder zu übertreffen [Vaj14], [Vaj15]. Die Ausprägung der Attribute kann vom Produktentwickler durch Veränderung der Merkmale und über die Partialmodelle ermittelten Eigenschaften beeinflusst werden. Dabei kann ein Merkmal bzw. eine Eigenschaft auch mehrere Attribute beeinflussen [VJK13]. In Abbildung 3.1 ist dies beispielhaft für den in der Größe hervorgehobenen gelben Produktparameter dargestellt. Dieser repräsentiert ein Merkmal, welches sich über die jeweiligen Eigenschaften auf fünf Attribute auswirkt.

Das Ziel der Produktentwicklung ist es, die Merkmale des Produktes so zu bestimmen, dass die geforderten Ausprägungen der Attribute und somit die Anforderungen an das Produkt erreicht oder in einigen Fällen auch übertroffen werden. Da zu Beginn der Produktentwicklung keine Aussagen über die exakten Wirkzusammenhänge zwischen Merkmalen und Attributen getroffen werden können, werden über die Partialmodelle zunächst die Eigenschaften des Produktes ermittelt. Aus diesen können anschließend die Ausprägungen der Attribute bestimmt werden. Zur Verbesserung der Attributsausprägung müssen anschließend die hierfür relevanten Merkmale identifiziert werden. Dies erfolgt durch Ermittlung der Wirkzusammenhänge.

Hierzu werden die Produktparameter mit Hilfe von *Tags* strukturiert. Ein Tag bezeichnet ein Anhängsel, Etikett oder Merker, welcher einem Objekt eine Meta- bzw. Zusatzinformation anfügt. Dadurch lassen sich Daten klassifizieren und strukturieren [FH11]. Neben den reinen Objektdaten können somit zusätzliche Informationen wie Ursprung, Verwendungszweck oder Versionsnummer gespeichert werden, ohne die eigentlichen Objektdaten zu verändern. Den Produktparametern können folgende Tags angefügt werden:

- Name
- Charakteristik
- Partialmodelle
- Attribute

Neben dem Namen wird die Charakteristik des Produktparameters erfasst. Dieser Tag beinhaltet die Information, ob der Produktparameter ein Merkmal oder eine Eigenschaft repräsentiert. Somit wird die Möglichkeit zur Veränderung der Charakteristik eines Parameters gewährleistet. Zusätzlich werden die Partialmodelle, in denen der Produktparameter verwendet wird, und die Attribute, die der Produktparameter beeinflusst, erfasst. Die Festlegung der Tags erfolgt zunächst manuell durch den Produktentwickler. Wie bereits

beschrieben, können dabei die Attribute, welche durch ein Merkmal beeinflusst werden, nicht ohne weitere Kenntnisse der Eigenschaften bestimmt werden. Den Merkmalen werden daher zunächst keine Attribut-Tags angefügt. Die Tags der Eigenschaften hingegen können vollständig festgelegt werden. Die Zuordnung der Attribute zu den Eigenschaften erfolgt ebenfalls durch den Produktentwickler.

Nach der Definition der Tags erfolgt die Zuordnung der Attribute zu den Merkmalen auf Basis der Partialmodelle. Diese stellen die Verbindung zwischen Merkmalen und Eigenschaften dar. Die Attribute der Eigenschaften werden dabei auf die Merkmale übertragen. Durch diesen Schritt, welcher automatisiert durch ein Programm ablaufen kann, werden die Wirkzusammenhänge zwischen Merkmalen und Attributen abgebildet.

Sind durch die Zuordnung die Attribut-Tags aller Produktparameter bekannt, können die Produktparameter der Matrix des Produktmodells hinzugefügt werden, wie in Abbildung 3.1 dargestellt ist. Die Matrixdarstellung dient somit auch als Entwurf einer Benutzungsoberfläche zur Analyse und Editierung des Produktmodells durch den Produktentwickler. Möchte der Produktentwickler die Ausprägung eines Attributes des Produktes verändern, ist nun schnell ersichtlich, welche Merkmale das Attribut beeinflussen und geändert werden können. Im Gegensatz zur in [VJK13] beschriebenen Tag-Variante, bei der dem Produktparameter die vollständige Wirkkette aus Partialmodell und Attribut als Tag angefügt wird, reduziert die hier beschriebene Variante den Modellierungsaufwand und die Fehleranfälligkeit bei der Modellierung, da die Eingabedaten einfacher sind.

Der folgende Quellcode zeigt beispielhaft die Verwendung von Tags in der Auszeichnungssprache XML für den in Abbildung 3.1 in der Größe hervorgehobenen Produktparameter.

```

1 <Produktparameter>
2   <Name>PP_Beispiel</Name>
3   <Charakteristik>Merkmal</Charakteristik>
4   <Partialmodelle>
5     <Partialmodell>Partialmodell 2</Partialmodell>
6     <Partialmodell>Partialmodell 3</Partialmodell>
7     <Partialmodell>Partialmodell 4</Partialmodell>
8   </Partialmodelle>
9   <Attribute>
10    <Attribut>Nachhaltigkeit</Attribut>
11    <Attribut>Sicherheit</Attribut>
12    <Attribut>Zuverlaessigkeit</Attribut>
13    <Attribut>Qualitaet</Attribut>
14    <Attribut>Rentabilitaet</Attribut>
15  </Attribute>
16 </Produktparameter>

```

Anhand der Tags ist erkennbar, dass das Merkmal *PP_Beispiel* in drei Partialmodellen verwendet wird und die Attribute Nachhaltigkeit, Sicherheit, Zuverlässigkeit, Qualität und Rentabilität beeinflusst. Alternativ zur Verwendung von Tags ist auch die Erstellung einer umfassenden Parameterdatenbank denkbar, in der alle relevanten Daten der Produktparameter sowie deren Wirkketten erfasst werden [TN16].

Durch die Identifizierung der Wirkketten zwischen Merkmalen und Eigenschaften bzw. zwischen Merkmalen und Attributen eines Produktes lassen sich die Merkmale gemäß ihrer Wirkung gruppieren. Diese Gruppierung erlaubt die Darstellung der Merkmale in einer dem biologischen Chromosom ähnlichen Form [VK10]. Der Umfang eines Chromosoms ist jedoch bei diesem chromosomähnlichen Produktbeschreibungsmodell noch nicht

genau spezifiziert. Denkbar ist die Zusammenfassung der Merkmale anhand der durch sie beeinflussten Eigenschaften oder Attribute. Die Definition eines Chromosoms auf Basis der Eigenschaften ist weniger komplex und somit einfacher handhabbar. Nachteilig bei diesem Ansatz ist die variable Anzahl an Chromosomen, da die Anzahl der Eigenschaften bei unterschiedlichen Produkten sowie über den Produktentwicklungsprozess variieren kann. Die Definition der Chromosomen auf Basis der Produktattribute ist zunächst komplexer in der Anwendung, bietet jedoch den Vorteil, dass die Anzahl der Chromosomen in jedem Fall konstant ist, da durch die Attribute im IDE eine allgemeingültige Beschreibung aller Produkte über den gesamte Produktlebenszyklus hinweg angestrebt wird.

Die Beschreibung des Produktes durch ein Chromosommodell auf Basis der Produktattribute bietet zudem den Vorteil der Unterscheidung zwischen dem Genotyp und dem Phänotyp eines Produktes analog zur biologischen Evolution (s. Abschnitt 3.1). Dabei bildet der Genotyp, welcher in der biologischen Evolution das Erscheinungsbild eines Chromosoms repräsentiert, auch im Chromosommodell das Erscheinungsbild eines Chromosoms ab. Dies entspricht den Merkmalen des Produktes. Der Phänotyp stellt in der biologischen Evolution das beobachtbare Erscheinungsbild eines Organismus dar. Im Chromosommodell entspricht der Phänotyp dem Erscheinungsbild des Produktes, das durch die Ausprägung der Attribute charakterisiert ist. Analog zur biologischen Evolution, in der der Phänotyp eines Organismus zudem durch Umwelteinflüsse geprägt wird, unterliegt auch der Phänotyp eines Produktes externen Einflüssen, die dessen Erscheinungsbild und Verhalten beeinflussen. Dies kann z. B. durch Einflüsse in der Fertigung oder während der Nutzung des Produktes erfolgen. Der Phänotyp eines Produktes stellt somit keinen permanenten Zustand dar, sondern ist über den Produktlebenszyklus veränderlich.

Durch die beschriebene Ermittlung der Wirkketten zwischen den Merkmalen und den Attributen eines Produktes sind zwischen diesen bisher lediglich die prinzipiellen Zusammenhänge bekannt. Es existieren jedoch keine Informationen über die quantitativen Einflüsse der Merkmale auf die Attribute. Einen Ansatz diese zu ermitteln bietet die explorative Analyse der Merkmale verbunden mit der Generierung von Metamodellen, welche die Einflüsse der einzelnen Merkmale (Sensitivitäten) abbilden (s. Abschnitt 2.7). Dies muss jedoch unter Verwendung der Eigenschaften erfolgen, da bisher keine quantitativ abbildbare Beschreibung zwischen Eigenschaften und Attributen existiert. Die Festlegung der Attribute erfolgt interaktiv durch den Produktentwickler auf Basis der Eigenschaften des Produktes.

In jedem Fall ist die Kenntnis der Sensitivitäten der Eigenschaften für den Produktentwickler sinnvoll, da er somit in der Lage ist, schnell die richtigen Merkmale auszuwählen, um eine Eigenschaft und dadurch ein Attribut gezielt zu verändern. Zudem können anhand der visualisierten Metamodelle, welche die Einflüsse der Merkmale auf die Eigenschaften abbilden, im Vorfeld die Konsequenzen einer Merkmalsänderung analysiert und insbesondere bei numerisch aufwendigen Partialmodellen der Berechnungsaufwand reduziert werden. Da die Informationen über die Sensitivitäten dem Produktentwickler idealerweise zu jedem Zeitpunkt der Produktentwicklung zur Verfügung stehen sollten, empfiehlt sich die Berechnung der zur Generierung der Metamodelle notwendigen Stützstellen im Hintergrund. Der damit verbundene Berechnungsaufwand kann durch die in dieser Arbeit entwickelte Parallelisierungsmethode in einem ausreichend großen Cluster verteilt werden. Bei der Verwendung von Metamodellen muss in jedem Fall berücksichtigt werden, dass diese lediglich eine Approximation des Verhaltens der Partialmodelle darstellen. Dies ist für die Bereitstellung der Sensitivitäten für den Produktentwickler zunächst ausreichend,

ersetzt jedoch nicht die exakte Bestimmung der Eigenschaften durch Evaluation der Partialmodelle. Dabei kann jede Evaluation eines Partialmodells als zusätzliche Stützstelle zur Verbesserung der Metamodelle genutzt werden.

Die Merkmale eines Produktes und somit der Genotyp werden während der Produktentwicklung iterativ verändert und angepasst, um die geforderte Ausprägung der Attribute zu erzielen. Die Veränderung der Merkmale kann manuell durch den Produktentwickler oder automatisiert durch den Einsatz von Optimierungsmethoden erfolgen. Folgende Ereignisse können zu Veränderungen der Merkmale führen [VK10]:

- Änderungen aufgrund von Abhängigkeiten: Die Veränderung eines Merkmals kann die Anpassung eines anderen Merkmals bedingen, da dieses nicht mehr alle vorher verfügbaren Werte annehmen kann. Wird z. B. über ein Merkmal ein bestimmtes Material verändert, können für ein Wandstärkemerkmale nicht mehr alle zuvor zulässigen Werte zulässig sein, wodurch eine Anpassung vorgenommen werden muss.
- Optimierung von Merkmalen: Dies stellt die häufigsten Veränderungen dar. Durch den Produktentwickler wird ein Merkmal angepasst, um eine Verbesserung einer Eigenschaft bzw. Attributsausprägung zu erzielen. Hierbei können auch Optimierungsmethoden zum Einsatz kommen.
- Externes Ereignis: Durch eine Änderung in den Anforderungen ergeben sich neue bzw. veränderte Tabuzonen und Restriktionen der Eigenschaften (s. Abschnitt 3.2.2), wodurch die Eigenschaften bestimmte Werte nicht mehr annehmen dürfen und die relevanten Merkmale verändert werden müssen.

Das Produktmodell der AKT zeichnet sich durch seine Dynamik und Flexibilität in der Anwendung aus. Die Produktparameter und die Partialmodelle können während der Produktentwicklung verändert und erweitert werden. Analog zum Stand der Produktentwicklung verändert sich auch das Produktmodell und erfährt eine Evolution. Mit jeder Evolutionsstufe wird das Produktmodell dabei umfassender und aussagekräftiger. Die Auswahl der Produktparameter und Partialmodelle richtet sich nach der Komplexität der zu lösenden Aufgaben und ist abhängig von den Methoden und Werkzeugen, die dem Produktentwickler zur Verfügung stehen. Auch die Granularität der Partialmodelle kann nicht generalisiert werden, sondern richtet sich ebenfalls nach der Komplexität der Entwicklungsaufgabe. Da im Verlauf der Produktentwicklung Informationen über das zu entwickelnde Produkt erarbeitet und spezifiziert werden, nimmt somit tendenziell die Anzahl sowie die Granularität der Partialmodelle zu. Die Notwendigkeit zur Verwendung eines bestimmten Partialmodells ergibt sich oft erst während der Bearbeitung einer konkreten Aufgabe, was auch auf die Attribute des Produktes zutrifft. Die Zweckmäßigkeit der Attribute sowie deren geforderte Ausprägung richtet sich dabei nach den Anforderungen an das zu entwickelnde Produkt. Da die Anforderungen oft während der Produktentwicklung erarbeitet und spezifiziert werden und sich zudem im Verlauf der Entwicklung ändern können, weisen auch die Attribute einen dynamischen Charakter auf [VJK13].

Das beschriebene Produktmodell liegt vollständig in digitaler Form vor und bietet somit Potential in der Auswertung relevanter Daten bezogen auf die verschiedenen Phasen und Aufgaben der Produktentwicklung. Insbesondere die Fragestellungen, in welchen Phasen und für welche Aufgaben die verschiedenen Partialmodelle genutzt werden und welche Produktparameter dabei involviert sind, sollten in der Zukunft näher untersucht werden, um somit den Produktentwickler durch eine gezielte Bereitstellung von Partialmodellen und Entscheidungshilfen zu unterstützen.

Weiterhin sollten die Zusammenhänge zwischen Eigenschaften und Attributen in quantifizierbarer Form ermittelt werden, um dadurch die Merkmale gezielt im richtigen Maß verändern zu können und somit die Anzahl von Iterationen zu reduzieren. Diese quantitative Ermittlung der Wirkzusammenhänge kann auf Basis von Metamodellen erfolgen, z. B. durch das Trainieren künstlicher neuronaler Netzwerke (s. Abschnitt 2.7.2). Durch die daraus resultierende vollständige digitale Erfassung der Wirkzusammenhänge zwischen den Merkmalen und den Attributen eines Produktes lassen sich parameterbasierte Optimierungsverfahren (s. Abschnitt 2.5.1) nutzen, um das Produkt gezielt auf bestimmte Attribute hin zu optimieren oder dem Produktentwickler Entscheidungsmodelle auf Basis Pareto-optimaler Lösungen (s. Abschnitt 2.8.1) bereitzustellen. Der mit diesem multi-kriteriellen und multidisziplinären Optimierungsansatz verbundene Berechnungsaufwand kann ebenfalls durch die in dieser Arbeit entwickelte Parallelisierungsmethode in einem ausreichend großen Cluster verteilt werden.

Ein wesentlicher Nachteil bei der Verwendung neuronaler Netzwerke zur Ermittlung der quantitativen Wirkzusammenhänge ist die große Anzahl an Stützstellen zum Trainieren des Netzwerkes. Es sind daher viele Iterationen nötig, in denen der Produktentwickler durch die Veränderung der Merkmale und auf Basis der Eigenschaften die Ausprägung der Attribute bewertet. Einen alternativen Ansatz bietet die in Abschnitt 3.2.3 vorgestellte Optimierung zur Konzeptfindung durch interaktive Evaluation, bei der die Bewertung der Attribute direkt innerhalb der Optimierung durch ein Entwicklungsteam erfolgt.

3.2.2 Lösungsraum

In der AKT beschreibt der Lösungsraum die Menge aller zulässigen Lösungselemente, die zur Generierung von Lösungen verwendet werden können. Er bildet alle Elemente ab, die der Produktentwickler zur Entwicklung bzw. Evolution einer oder mehrerer Lösungen verwenden kann. Dies beinhaltet die gestellten Anforderungen, Anfangsbedingungen, interne und externe Randbedingungen, Restriktionen sowie sämtliche weitere geltenden Bedingungen. Da sich diese im Verlauf einer Produktentwicklung dynamisch verändern können, besitzt auch der Lösungsraum dynamischen Charakter und kann sich im Verlauf der Produktentwicklung verändern [VK10].

Der Lösungsraum der AKT kann mit dem Definitionsbereich einer mathematischen Funktion verglichen werden [VK10]. Er bildet somit die definierenden Eigenschaften eines Systems bzw. eines Produktes ab, welche auch als Merkmale bezeichnet werden (s. Abschnitt 3.2.1). Diese Definition widerspricht der in dieser Arbeit verwendeten Definition des Lösungsraumes einer Optimierung (s. Abschnitt 2.4). Dieser basiert auf den Zustandsvariablen und spiegelt die beschreibenden Eigenschaften des zu optimierenden Objektes wider. Der Lösungsraum einer Optimierung repräsentiert den Wertebereich der Zielfunktion und dient der Auswahl einer oder mehrerer Pareto-optimaler Lösungen. Die definierenden Eigenschaften und somit der Definitionsbereich der Zielfunktion werden durch den Designvariablenraum gebildet. Da der Designvariablen- und der Lösungsraum einer Optimierung über die Zielfunktion bzw. das Evaluationsmodell gegeneinander abgebildet werden können und somit die Transformation von Lösungen zwischen beiden Räumen möglich ist, kann die Diskrepanz der Begriffe von AKT und Optimierungsprozess in diesem Abschnitt vernachlässigt werden. Daher wird in diesem Abschnitt die Lösungsraumdefinition der AKT verwendet.

Im Verlauf einer Produktentwicklung erfährt der Lösungsraum eine permanente Veränderung und Anpassung. Dabei wird er zunächst ausschließlich durch die geltenden Naturgesetze begrenzt und beinhaltet alle möglichen Elemente zur Lösung einer Aufgabe. Da bei der Entwicklung eines Produktes nur begrenzte Ressourcen zur Verfügung stehen, kann nur eine begrenzte Anzahl an Varianten bzw. Lösungselementen untersucht werden und einige Lösungselemente bleiben unberücksichtigt. Zudem entstehen durch Restriktionen Tabuzonen, welche den Lösungsraum weiter einschränken. Die Einschränkungen resultieren in einer Menge möglicher Lösungen, die zur Lösungsfindung genutzt werden. Diese Menge wird als Entwicklungskorridor bezeichnet. Abbildung 3.2 zeigt die schematische Darstellung des Lösungsraumes der AKT.

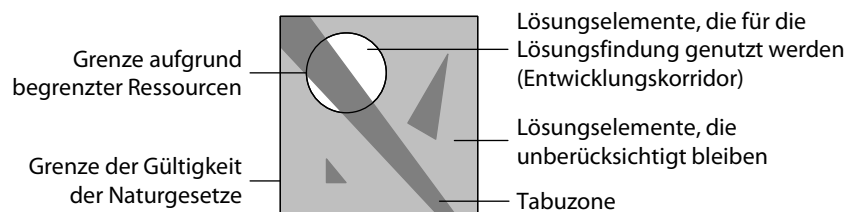


Abbildung 3.2: Grenzen und Elemente des Lösungsraumes der AKT, nach [VJK13]

Neben der dynamischen Veränderung der die Lösungselemente begrenzenden Objekte ist der Produktentwicklungsprozess durch ein stetiges Generieren und Auswählen von Lösungsvarianten gekennzeichnet. Durch den Produktentwickler werden dabei in der Regel zunächst viele verschiedene mögliche Lösungen generiert, welche anschließend bezogen auf die Problemstellung wieder zusammengeführt und fokussiert werden. Dies wird auch als Divergenz und Konvergenz von Lösungen bezeichnet [Ma07]. Die Konvergenz der Lösungen kann dabei in einer oder mehreren Lösungen resultieren. Die Auswahl der Lösungen erfolgt zudem unter Berücksichtigung der zur weiteren Detaillierung der Lösungen zur Verfügung stehenden Ressourcen. Nach der Auswahl werden die Lösungen durch den Produktentwickler weiterentwickelt, wobei ebenfalls wieder Varianten generiert und bewertet bzw. ausgewählt werden. Durch die Bildung verschiedener Varianten in jedem Entwicklungsschritt erfolgt eine Expansion des Lösungsraumes (Divergenz). Die Entscheidungen des Produktentwicklers bei der Auswahl von Lösungen verkleinern den Lösungsraum hingegen (Konvergenz). Daher wird auch der Begriff des *atmenden Lösungsraumes* verwendet. Ohne die Einschränkung nach jedem Entwicklungsschritt würde sich die Anzahl der Varianten, welche im nächsten Schritt detailliert werden, exponentiell vergrößern, wodurch die Komplexität des Entwicklungsprozesses stark ansteigt und die Handhabbarkeit gefährdet wird [VJK13]. Der atmende Lösungsraum ist somit eine Voraussetzung für die Generierung innovativer Lösungen durch Kreativität, aber auch zur Gewährleistung eines stabilen und handhabbaren Produktentwicklungsprozesses.

Durch die Auswahl einer bestimmten Anzahl an Lösungen durch den Produktentwickler, welche als gleichwertig beschrieben werden können, entsteht im Lösungsraum ein Korridor mit Lösungselementen. Dieser wird als Entwicklungskorridor bezeichnet. Ist der Entwicklungskorridor frei von Tabuzonen, korreliert dessen Größe mit den zur Verfügung stehenden Ressourcen. Die Form des Entwicklungskorridors sowie dessen Lage im Lösungsraum ergeben sich ebenfalls aus den Entscheidungen des Produktentwicklers. Da sich sowohl die Ressourcen als auch die Randbedingungen im Verlauf der Produktentwicklung verändern, ist auch der Entwicklungskorridor veränderlich. Hinzu kommen Tabuzonen aufgrund von

Restriktionen, welche den Lösungsraum sowie den Entwicklungskorridor einschränken und sich ebenfalls verändern können [VJK13]. Die schematische Entwicklung des Lösungsraumes während der Produktentwicklung ist in Abbildung 3.3 dargestellt.

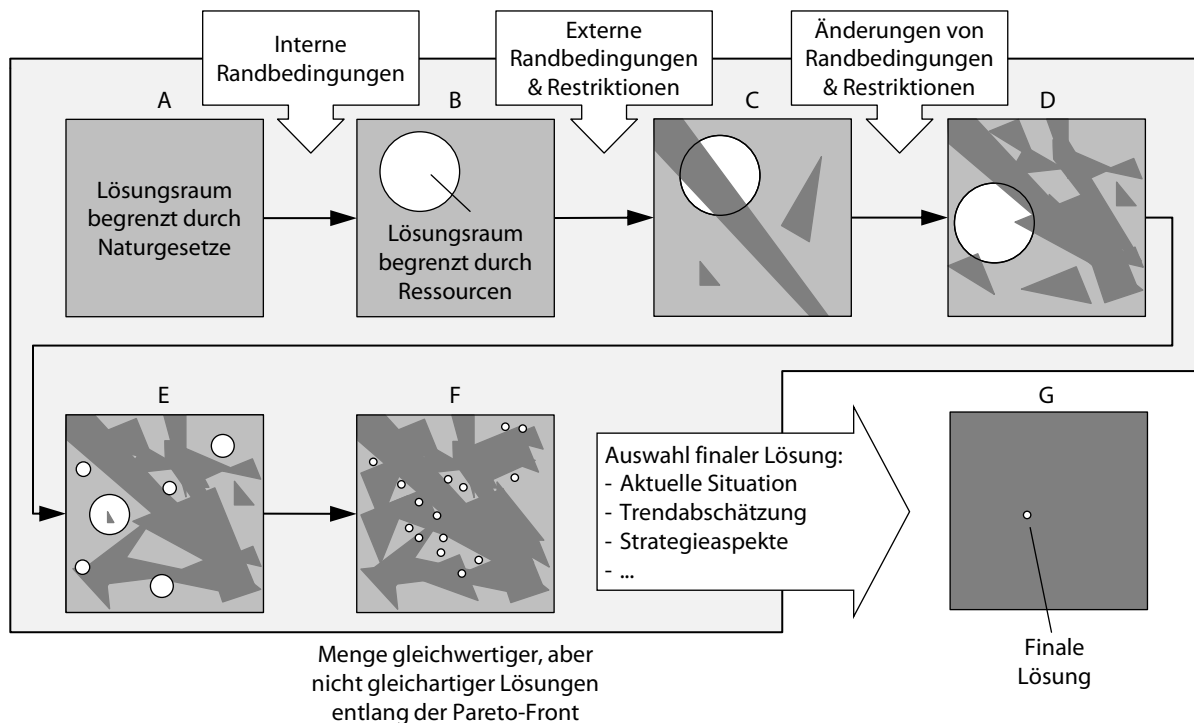


Abbildung 3.3: Schematische Entwicklung des Lösungsraumes der AKT bei der Produktentwicklung, nach [VJK13]

Die Entwicklung des Lösungsraumes folgt in der AKT keinem starren Schema, sondern ist flexibel anpassbar und hochgradig dynamisch. Diese Eigenschaften finden sich auch in der biologischen Evolution wieder (s. Abschnitt 3.1). Der in Abbildung 3.3 dargestellte sequentielle Ablauf dient lediglich der Erläuterung der prinzipiellen Schritte. Diese sind in der Reihenfolge flexibel auf die jeweilige Problemstellung anpassbar. Zudem sind in jedem Schritt Iterationen möglich. Auf eine Darstellung des atmenden Lösungsraumes durch Divergenz und Konvergenz von Lösungen, welche prinzipiell in jedem Schritt erfolgen können, wird aufgrund der Übersichtlichkeit verzichtet.

Sind zu Beginn der Produktentwicklung keine Anforderungen definiert, wird der Lösungsraum allein durch die Naturgesetze begrenzt (A). In den meisten Fällen existieren zu Beginn nur wenige und grob formulierte Anforderungen. Der Lösungsraum ist somit nicht bzw. kaum durch Tabuzonen eingeschränkt, woraus eine Vielzahl möglicher Lösungen resultiert. Aufgrund begrenzter Ressourcen kann der Produktentwickler jedoch nur einen Teil dieser Lösungen berücksichtigen (B). Zu diesen internen Randbedingungen gehören z. B. Mitarbeiterkapazitäten, innerbetriebliche Strukturen und Abläufe sowie Kenntnisse und Verfügbarkeit von Methoden und Werkzeugen. Durch die Erarbeitung und Konkretisierung der Anforderungen an das zu entwickelnde Produkt entstehen externe Randbedingungen und Restriktionen, welche invertiert durch Tabuzonen beschrieben werden (C). Diese stellen die nicht zulässigen Bereiche des Lösungsraumes dar. Die Summe aller Tabuzonen bildet der Verbotraum, welcher den vollständigen invertierten Lösungsraum repräsentiert. Im Verlauf der Produktentwicklung trifft der Produktentwickler eine Vielzahl

von Entscheidungen, welche die Tabuzonen sowie die Form und Lage des Entwicklungskorridors beeinflussen und verändern (D). Mit der Definition des Entwicklungskorridors erfolgt indirekt eine Beschreibung des Produktmodells, da dieses die Merkmale des Produktes und deren Varianten abbildet (s. Abschnitt 3.2.1). Das Produktmodell setzt sich somit aus den Elementen des Lösungsraumes zusammen, welche sich innerhalb des Entwicklungskorridors befinden [VJK13].

Im Idealfall befinden sich innerhalb des Entwicklungskorridors die Lösungen, welche die optimale Antwort auf die Entwicklungsaufgabe darstellen. Da die Entwicklung eines Produktes aufgrund der Vielzahl von Anforderungen in der Regel einen multikriteriellen Charakter aufweist und mit einer multikriteriellen Optimierung verglichen werden kann, beinhaltet die optimale Antwort auf die Entwicklungsaufgabe eine Pareto-optimale Menge möglicher Lösungen. Diese Lösungen werden nicht durch andere Lösungen dominiert und sind somit gleichwertig, jedoch nicht gleichartig (s. Abschnitt 2.8.1).

Kommt es im Verlauf der Produktentwicklung zu einer Änderung der Anforderungen, verändert sich neben dem Lösungsraum auch die Bewertung der Lösungselemente. Dies kann zur Folge haben, dass eine oder mehrere optimale Lösungen nicht generiert werden können, da deren Lösungselemente sich nicht im Entwicklungskorridor und somit außerhalb des Blickwinkels des Produktentwicklers befinden. In der AKT werden derartige Problemstellungen analog zur biologischen Evolution betrachtet. Ein geringer Teil der Lösungsvarianten wird durch *Mutation* verändert und eröffnet der Population somit neue zulässige Bereiche des Lösungsraumes. Die Mutation der Lösungen führt zu einer Teilung des Entwicklungskorridors (E). Dieser Prozess wirkt einem Determinismus entgegen und verhindert, dass die Entwicklung als Ganzes in einer Lösung konvergiert. Dabei wird kontinuierlich eine bestimmte Anzahl gleichwertiger Lösungen vorgehalten [VJK13].

Während der Entwicklung besteht die primäre Aufgabe des Produktentwicklers darin, unter Berücksichtigung der sich verändernden Tabuzonen den Entwicklungskorridor so zu gestalten, dass am Ende der Produktentwicklung eine bestimmte Anzahl Pareto-optimaler Lösungen existiert (F). Aus diesen Lösungen wählt der Produktentwickler die finale Lösung aus, welche letztendlich das Ergebnis der Produktentwicklung darstellt und in einem realen Produkt realisiert wird (G). Bei der Auswahl können z. B. die generelle aktuelle Situation in einem Bereich oder einer Organisation, Abschätzungen von Trends sowie strategische Aspekte die Entscheidung beeinflussen. In diesem letzten Schritt der Produktentwicklung ist der Lösungsraum soweit konkretisiert und eingeschränkt, dass dieser bis auf die finale Lösung vom gesamten Verbotsraum abgedeckt wird [VJK13].

In jedem der beschriebenen Entwicklungsschritte, in denen Lösungen generiert und ausgewählt werden, können je nach Aufgabenstellung verschiedene Optimierungsmethoden und -verfahren zum Einsatz kommen und den Produktentwickler bei der Generierung Pareto-optimaler Lösungen unterstützen. Da bei der Verwendung von Optimierungsmethoden und -verfahren in der Regel mehr Lösungsvarianten untersucht werden, als manuell durch den Produktentwickler, ist auch diese Anwendung kennzeichnend für den atmenden Charakter des Lösungsraumes. Die in dieser Arbeit entwickelten Parallelisierungsmethoden und deren prototypische Umsetzung liefern einen Beitrag zum effizienten Einsatz multikriterieller und multidisziplinärer Optimierungen in dem beschriebenen Umfeld der Produktentwicklung, insbesondere wenn diese den Einsatz numerisch aufwendiger Berechnungsmethoden bedingen.

3.2.3 Anwendung der AKT in den frühen Phasen der Produktentwicklung

Eine Form der Anwendung der AKT beinhaltet den Einsatz verschiedener Optimierungsstrategien unter Verwendung evolutionärer und genetischer Algorithmen. Dies kann durch das Optimierungssystem NOA erfolgen, welches verschiedene Module evolutionärer und genetischer Algorithmen beinhaltet [JC04]. Dabei bedingen bisherige Anwendungen jedoch stets ein Evaluationsmodell zur Beschreibung der Produkteigenschaften, welches vollständig digital erfasst sein muss, um es in einem automatisierten Optimierungsprozess nutzen zu können. Insbesondere in den frühen Phasen der Produktentwicklung, in denen verschiedene Konzepte eines Produktes erarbeitet und bewertet werden, welche die Grundlage für die weitere Entwicklung eines Produktes darstellen, existieren oft noch keine aussagekräftigen Modelle, welche das Produktverhalten ausreichend genau beschreiben. Die Generierung und Auswahl von Konzepten erfolgt in diesen Phasen meist auf der Basis von Bildern, Skizzen und Gedankenmodellen, welche anschließend bewertet werden.

Eine Methode, welche sich zur Unterstützung der Konzeptentwicklung etabliert hat, ist der *Morphologische Kasten*. Dabei wird das Gesamtproblem zunächst in Teilprobleme unterteilt, für die im nächsten Schritt unterschiedliche Teillösungen erarbeitet werden. Die Visualisierung der Teilprobleme und -lösungen erfolgt in einer Matrixdarstellung, in der anschließend die jeweiligen Teillösungen kombiniert und somit Lösungskonzepte für das Gesamtproblem erarbeitet werden [Zwi66]. Die Anwendung des Morphologischen Kastens kann dabei auf verschiedenen Abstraktions- und Detaillierungsstufen eines Problems erfolgen. Die Generierung der Konzepte stellt ein kombinatorisches Problem dar, bei dem die Anzahl der möglichen Lösungen exponentiell mit der Anzahl der Teilprobleme ansteigt. Bereits ein Morphologischer Kasten bestehend aus fünf Teilproblemen mit jeweils fünf Teillösungen führt zu 3125 möglichen Lösungen des Gesamtproblems, welche nicht alle durch den Produktentwickler evaluiert werden können z. B. aufgrund begrenzter Ressourcen (s. Abschnitt 3.2.2).

Aus dieser Motivation und unter Berücksichtigung der in Abschnitt 3.1 aufgeführten Eigenschaften der biologischen Evolution wurde das Konzeptoptimierungssystem *Morphix* entwickelt. Dieses basiert auf der Synthese aus Morphologischem Kasten und AKT und unterstützt den Produktentwickler bei der Generierung und Auswahl von Produktkonzepten durch eine im Hintergrund laufende Optimierung [WPV16]. Die Definition der Designvariablen erfolgt implizit durch das System auf Basis des vom Produktentwickler erstellten Morphologischen Kastens. Morphix repräsentiert somit eine Methode zur Konzeptoptimierung mit impliziter Definition der Designvariablen (s. Abschnitt 2.1). Dabei kommt ein Genetischer Algorithmus (GA) des Optimierungssystems NOA zum Einsatz, da sich insbesondere GA zur Lösung komplexer Probleme mit un stetigen Zielfunktionen sowie für kombinatorische und diskrete Optimierungsprobleme eignen, wie sie auch bei der Konzeptentwicklung auftreten (s. Abschnitt 2.6.2.1).

Hingegen konventioneller Anwendungen von GA zur Optimierung von Produkten, bei denen das Evaluationsmodell in digitaler Form vorliegen muss, erfolgt die Evaluation der Konzepte interaktiv durch ein interdisziplinäres Entwicklungsteam. Die Konzepte werden dabei lediglich durch Bilder der jeweiligen Teillösungen repräsentiert und durch die Teammitglieder bewertet. Bei der Evaluation der Konzepte werden direkt die *Attribute* des IDE verwendet [Vaj14]. Diese repräsentieren die Anforderungen an das zu entwickelnde Produkt und finden bereits im Produktmodell der AKT Anwendung (s. Abschnitt 3.2.1).

Eine Bestimmung der konkreten Eigenschaften der Konzepte findet nicht statt, da diese aufgrund fehlender aussagekräftiger Modelle in dieser frühen Entwicklungsphase noch nicht sicher bestimmt werden können. Zudem ermöglicht die Verwendung der Attribute die einfache Überführung der ausgewählten Konzepte in das Produktmodell, sowie den Abgleich der Attribute über verschiedene Evolutionsstufen des Produktes hinweg. Das in Morphix verwendete interaktive Evaluationsmodell zur Konzeptauswahl repräsentiert somit ein sehr grobgranulares, frühes Partialmodell im Produktmodell der AKT.

Zur Unterstützung der Konzeptevaluation auf Basis der Attribute besitzt jedes Attribut verschiedene Subattribute, welche es näher beschreiben und den Mitgliedern des Entwicklungsteams als Orientierung dienen. Eine Übersicht der in Morphix integrierten Subattribute wird in [WPV16] gegeben. Bezogen auf die Aufgabenstellung werden die Attribute bzw. Subattribute im Vorfeld der Konzeptevaluation ausgewählt und mittels Gewichtungsfaktoren die Relevanz der Subattribute festgelegt. Die Subattribute stellen somit die Bewertungskriterien bei der Evaluation dar. Zusätzlich können eigene Bewertungskriterien erstellt werden. Durch die Gewichtung der Kriterien wird die multikriterielle Optimierungsaufgabe auf eine monokriterielle Zielfunktionsformulierung reduziert (s. Abschnitt 2.8.1.1). Dies entspricht der konventionellen Bewertung von Varianten auf Basis von Gewichtungsfaktoren und Bewertungskriterien in der Produktentwicklung [FG13]. Die Fitnesszuweisung der evaluierten Konzepte erfolgt anschließend proportional aus den Zielfunktionswerten (s. Abschnitt 2.6.2.1). Abbildung 3.4 zeigt die Benutzungsoberfläche von Morphix zur interaktiven Konzeptevaluation am Beispiel eines 3D-Druckers.

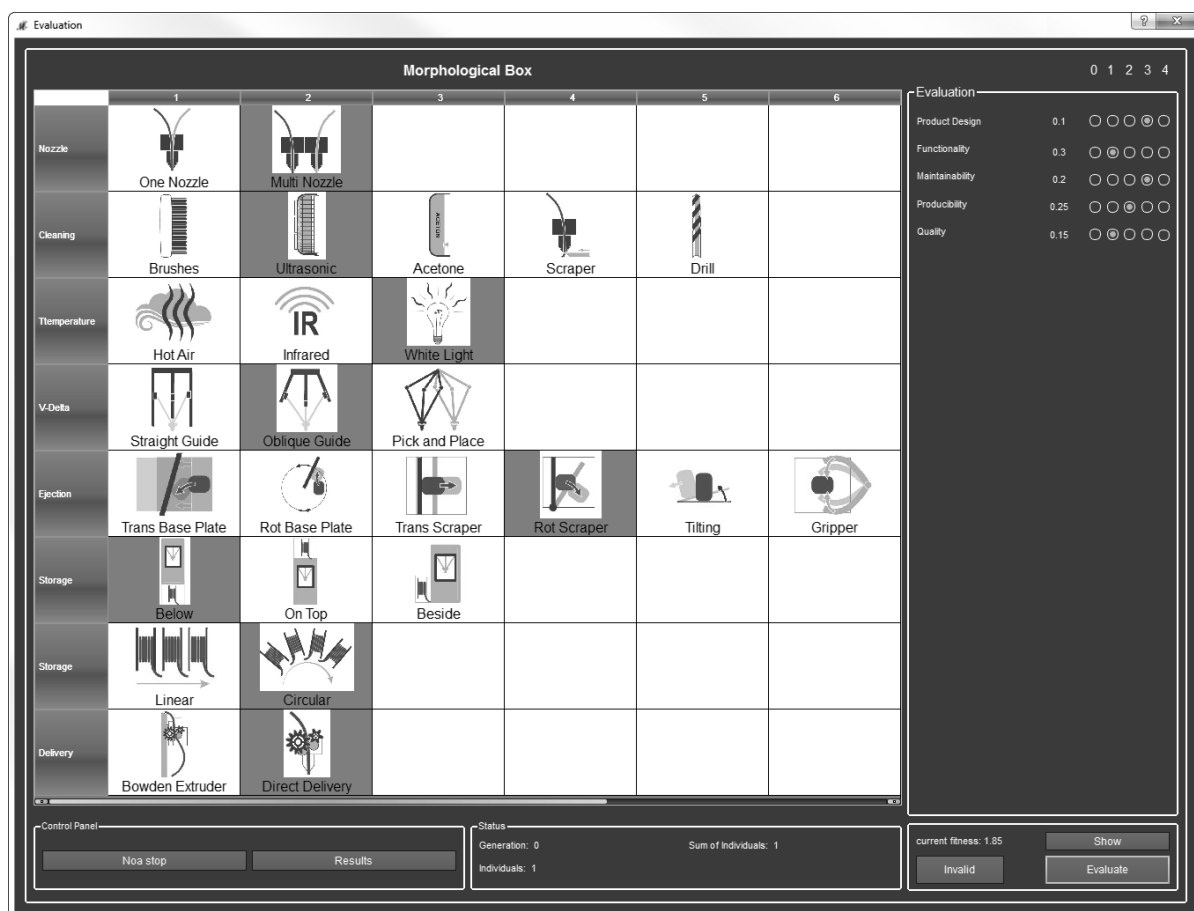


Abbildung 3.4: Benutzungsoberfläche von Morphix zur interaktiven Konzeptevaluation [WPV16]

Die Darstellung des Morphologischen Kastens bildet das zentrale Element der Benutzungsoberfläche während der Konzeptevaluation. Das zu evaluierende Konzept wird durch die hervorgehobenen Teillösungen repräsentiert. Die Evaluation der Konzepte erfolgt im rechten Bereich der Benutzungsoberfläche. In diesem Bereich werden die verwendeten Attribute sowie die dazugehörigen Gewichtungsfaktoren aufgeführt. Durch das Entwicklungsteam können Werte von 0 – 4 vergeben werden. Ein Wert von 0 repräsentiert keine Ausprägung, 4 eine sehr starke Ausprägung eines Attributes. Hieraus erfolgt die Berechnung des Zielfunktionswertes, welcher aufgrund der proportionalen Zuweisung auch die Fitness und somit die Selektionswahrscheinlichkeit repräsentiert. Das Ziel der Optimierung ist die Maximierung der Fitness.

Die Evaluation der Konzepte beginnt mit der zufälligen Generierung der Initialgeneration durch den GA. Während der Evaluation besteht stets die Möglichkeit, die besten drei Konzepte aufzurufen und somit das aktuell zu bewertende Konzept mit diesen zu vergleichen. Im Verlauf der Optimierung werden die besten Teillösungen sukzessive durch den GA identifiziert und miteinander kombiniert (Rekombination). Zudem erfolgt eine Mutation der Konzepte bzw. Individuen, um innerhalb der Population die Diversität der Individuen zu gewährleisten (s. Abschnitt 2.6.2.1). Werden durch den GA Konzepte generiert, welche nicht realisierbar sind, können diese als ungültig (*invalid*) markiert werden und erhalten somit einen Fitnesswert $f_{Fit} = 0$. Ungültige Konzepte werden bei der Selektion folglich nicht berücksichtigt und nicht durch den GA weiterverfolgt.

Bei der in Abbildung 3.4 dargestellten Konzeptgenerierung zur Entwicklung eines 3D-Druckers wurde eine Populationsgröße $n_I = 10$ in Kombination mit zwei Abbruchkriterien verwendet. Die Optimierung sollte beendet werden, wenn über fünf Generationen keine Verbesserung des Fitnesswertes erzielt wurde. Die maximale Anzahl an Generationen wurde auf 15 festgelegt. Dies führte zu 144 durch das Entwicklungsteam evaluierten Konzepten, von denen 30 als ungültig bewertet wurden. Ausgehend von 6480 möglichen Lösungen wurden dabei nur 2,2% der möglichen Lösungen evaluiert. Das Entwicklungsteam benötigte hierzu zwei Meetings mit einer Dauer von jeweils 90 min. Obwohl die Anwendung von Morphix zu einer Bestätigung der im Vorfeld getroffenen Konzeptauswahl führte und dabei gleichermaßen auch alternative, gleichwertige Konzepte generiert wurden, ist die hohe Anzahl an Evaluationen und die damit verbundene Zeit als nachteilig zu bewerten. Dies erschwert die Anwendung bei komplexeren Produkten. Analog zur konventionellen Anwendung eines Morphologischen Kastens muss der Grad der Abstraktion und Detaillierung sorgfältig vom Produktentwickler festgelegt werden. Eine detaillierte Beschreibung der Funktionsweise und der Anwendung von Morphix in verschiedenen Fallstudien sowie eine ausführliche Diskussion werden in [WPV16] gegeben.

Die in diesem Abschnitt beschriebene evolutionäre Methode zur Unterstützung der Konzeptentwicklung basierend auf interaktiver Evaluation bietet in verschiedenen Richtungen Potential für weitere Untersuchungen. Obwohl die Evaluation der Konzepte interaktiv durch die Teammitglieder erfolgt, wird die Konzeptentwicklung in digitaler Form beschrieben. Die damit verbundenen Daten können somit für weitere Untersuchungen der Produktentwicklung und des Produktmodells genutzt werden.

Die Evaluation der Konzepte basiert auf den Attributen des IDE, welche ebenfalls in das Produktmodell der AKT integriert sind. Dies ermöglicht eine einfache Überführung des interaktiven Modells in das Produktmodell und bietet Potential zur weiteren Erforschung des Produktentwicklungsprozesses, z. B. hinsichtlich der Veränderung der Attributsausprägung von ausgewählten Konzepten im Verlauf der weiteren Detaillierung oder

in der Untersuchung der Ursachen ungewollter Iterationen, verursacht durch eine suboptimale Konzeptauswahl. Zudem lassen sich aufgrund der digitalisierten Konzeptentwicklung problemspezifische Teillösungen in einer Datenbank speichern und durch die Bereitstellung dieser Daten zukünftige Konzeptentwicklungen bei ähnlichen Teilproblemen unterstützen.

Weiterhin sollte der Evaluationsprozess der Konzepte näher untersucht und ggf. vereinfacht werden. So ist z. B. die automatische Erkennung ungültiger Konzepte auf Basis von Erfahrungswerten und Metadaten denkbar. Auch die Implementierung alternativer Verfahren zur Evaluation bietet weiteres Potential. So erlaubt die Verwendung multi-kriterieller GA, bei denen die Fitnesszuweisung rangbasiert auf Basis nicht dominierter Pareto-optimaler Lösungen erfolgt, die Evaluation rein auf den Zielkriterien, ohne dass Gewichtungsfaktoren festgelegt werden müssen (s. Abschnitt 2.8.1.4). Jedes Zielkriterium ist somit gleichwertig. Diese Form der Bewertung bildet eine Möglichkeit, die Gleichwertigkeit der Attribute herauszustellen. Da der Produktentwickler auch bei diesen Evaluationsverfahren möglichst objektiv entscheiden muss, stellt die Implementierung einer binären Evaluation eine weitere Möglichkeit zur Vereinfachung des Evaluationsprozesses dar. Dabei würde der Produktentwickler bzw. das Entwicklungsteam bei jeder Evaluation ausschließlich zwei Konzepte hinsichtlich der Attribute miteinander vergleichen. Die Berechnung der Fitness würde im Hintergrund erfolgen [WPV16].

Zudem zeigt sich bei der konventionellen Anwendung des Morphologischen Kastens oft, dass die Produktentwickler bereits im Vorfeld bestimmte Konzepte favorisieren. Bei der Verwendung eines GA werden im Vorfeld hingegen keine Präferenzen berücksichtigt. Die Erzeugung der Initialgeneration erfolgt in der Regel rein zufällig. Ausgehend von der Initialgeneration werden zielführende Gene identifiziert und über Generationen hinweg miteinander kombiniert und mutiert (s. Abschnitt 2.6.2.1). Zur Berücksichtigung der vom Produktentwickler favorisierten Konzepte ist es denkbar, neben den zufällig generierten Individuen auch die favorisierten Konzepte als Individuen zu initiieren, indem diese vom Produktentwickler vorgegeben werden. Gute Teillösungen würden somit womöglich schneller durch den GA erkannt und miteinander kombiniert werden können.

Da bei der interaktiven Evaluation die menschliche Komponente das limitierende Element darstellt, führt die in dieser Arbeit entwickelte Parallelisierungsmethode nicht direkt zu einer Effizienzsteigerung der Optimierung. Nach dem aktuellen Stand stellt die beschriebene Konzeptentwicklung eine multikriterielle und monodisziplinäre Optimierung dar, da nur ein interaktives Evaluationsmodell verwendet wird. Eine Effizienzsteigerung durch Parallelisierung ist jedoch denkbar, indem der Evaluationsprozess der Konzeptvarianten nicht zentralisiert durch ein Team stattfindet, sondern auf die Teammitglieder verteilt wird. Besitzt jedes Teammitglied eine spezifische fachliche Expertise bezogen auf das zu entwickelnde Produkt und bewertet ausschließlich eigene fachspezifische Zielkriterien, bekommt die Evaluation einen multidisziplinären Charakter und kann ebenfalls parallel sowie ggf. auch dezentral erfolgen. Jedoch verringert sich durch die fehlende Kommunikation zwischen den Teammitgliedern auch die Möglichkeit der Lösungsverbesserung durch Diskussionen. In jedem Fall erfordert der beschriebene Ansatz zur Anwendung der AKT in den frühen Phasen der Produktentwicklung durch interaktive Evaluation weitere Erforschung.

4 Effizienzsteigerung von Optimierungsverfahren durch Parallelisierung

In diesem Kapitel wird die effiziente Bearbeitung von Optimierungsaufgaben in der Produktentwicklung durch dynamische Parallelisierung der erforderlichen Evaluationsprozesse beschrieben. Ausgehend von den theoretischen Grundlagen der Modelldekomposition und der parallelen Bearbeitung werden zwei grundsätzliche Formen der Modelldekomposition und der Parallelisierung von Evaluationsprozessen in der Optimierung vorgestellt. Es erfolgt eine Beschreibung der Anwendung konventioneller Parallelisierungsmethoden, wobei besonderer Fokus auf der Anwendung Genetischer Algorithmen liegt, da diese eine hohe Wahrscheinlichkeit liefern, das globale Optimum eines Problems zu finden und da sie von ihrer grundlegenden Struktur bereits parallel arbeiten.

Die beschriebenen konventionellen Parallelisierungsmethoden berücksichtigen nur selten die verwendeten Ressourcen sowie die Limitierung dieser Ressourcen. Die vorhandene Software und die zur Verfügung stehenden Softwarelizenzen finden keine Berücksichtigung. Da zur Evaluation der Produktvarianten bei einer Optimierung in der Regel kommerzielle Software verwendet wird, deren Verfügbarkeit den Grad der Parallelisierung und somit die Effizienz der Optimierung einschränken kann, wird die konventionelle Ressourcendefinition erweitert und unter den Begriffen Concurrent Optimization und Simultaneous Optimization eine neue Methode zur effizienten Parallelisierung von Evaluationsprozessen von Optimierungen in der Produktentwicklung vorgestellt. Diese basiert auf Analogien von Produktentwicklungs- und Optimierungsprozessen sowie den Charakteristiken der Methoden des Concurrent und des Simultaneous Engineering.

Da die vorgestellte Methode zur Parallelisierung eine verteilte IT-Umgebung erfordert, werden die wesentlichen Eigenschaften des verteilten Rechnens erläutert. Besonderer Fokus liegt hierbei auf der Verwendung von Workstation-Clustern. Auf Basis von erarbeiteten Anforderungen an die verteilte IT-Umgebung wird ein Cluster-Management-System zur Realisierung des Clusters ausgewählt. Abschließend wird die prioritätenbasierte Bearbeitung von Berechnungsaufgaben in einem Cluster beschrieben und ein Überblick über die Methoden zur Ermittlung der Prozessprioritäten gegeben.

4.1 Parallelisierung von Optimierungsverfahren durch Modelldekomposition

Die Eignung eines Problems zur Parallelisierung richtet sich stark nach dessen Struktur. Besteht das Problem aus voneinander unabhängigen Teilproblemen, wird dies als *inhärenter Parallelismus* bezeichnet. Das Problem kann in seine Teilprobleme zerlegt und diese unabhängig voneinander verarbeitet werden. Diese Zerlegung wird als Dekomposition bezeichnet [BBKS15]. Da die Verarbeitung der Teilprobleme auf verschiedenartigen Ressourcen wie CPU oder ganzen Rechnern möglich ist, werden diese folgend als *Slots*

bezeichnet. Ein Slot stellt die kleinste Einheit einer Ressource dar, welche nicht weiter unterteilt werden kann.

Maßgebend für die Bewertung paralleler Verarbeitung ist die Laufzeit t . Diese setzt sich aus der Bearbeitungszeit t_{work} , der Kommunikationszeit t_{com} , der Wartezeit t_{wait} und der Synchronisationszeit t_{syn} zusammen [BBKS15]:

$$t = t_{work} + t_{com} + t_{wait} + t_{syn} \quad (4.1)$$

Dabei können Kommunikations-, Warte- und Synchronisationszeit zur Overhead-Zeit t_{over} zusammengefasst werden:

$$t_{over} = t_{com} + t_{wait} + t_{syn} \quad (4.2)$$

Ein Problem wird am effizientesten bearbeitet, wenn die Anzahl der Slots n_s der Anzahl der Teilprobleme n_p oder einem ganzzahligen Teil von diesen entspricht und eine vollständig parallele Verarbeitung erfolgt. Inhärent parallele Probleme können fast ohne jede Kommunikation zwischen den einzelnen Teilproblemen bearbeitet werden. Kommunikation ist lediglich vor und nach der Bearbeitung nötig, da die Daten im Vorfeld an die Slots übermittelt und nach Fertigstellung die Ergebnisse von den Slots übermittelt werden müssen. Somit ist die Zeiteinsparung nahezu linear zum Grad der Zerlegung. Diese lineare Zeiteinsparung wird als linearer Speedup bezeichnet [BM06].

Der Speedup S_p stellt ein Maß für die Reduktion der Laufzeit des Gesamtproblems durch die Parallelisierung dar. Er spiegelt somit den Grad der Parallelisierung wider und berechnet sich aus der schnellsten sequentiellen Laufzeit t_{seq} und der parallelen Laufzeit t_p durch den Einsatz paralleler Slots:

$$S_p = \frac{t_{seq}}{t_p} \quad (4.3)$$

In der Regel ist der Speedup durch die Anzahl der Slots n_s limitiert ($S_p \leq n_s$). Entspricht der erreichte Speedup der Anzahl an Slots ($S_p = n_s$), liegt linearer Speedup vor. Dieser Idealzustand sollte in der Regel angestrebt werden. In einigen Fällen ist sogar das Erreichen von superlinearem Speedup möglich ($S_p > n_s$), z. B. bei der Parallelisierung von Monte-Carlo-Simulationen. In diesem Fall ist der parallele Algorithmus effizienter als die beste bekannte sequentielle Variante. Dieses Verhalten stellt jedoch eher eine Ausnahme dar [BBKS15].

Ein weiteres Maß für die Beurteilung der relativen Verbesserung der Verarbeitungsgeschwindigkeit ist die Effizienz E_p . Hierbei wird der erreichte Speedup S_p auf die Anzahl der Slots n_s normiert:

$$E_p = \frac{S_p}{n_s} \quad (4.4)$$

Die Effizienz gibt an, wie effizient das ursprünglich sequentielle Problem parallelisiert und wie effizient die Slots für die Bearbeitung ausgenutzt werden. Eine Effizienz gleich 1 repräsentiert den Idealzustand, der angestrebt werden sollte. In der Regel liegt die Effizienz jedoch darunter. Der Grund für diese Differenz liegt in den benötigten Zeiten für Kommunikation, Wartevorgänge und Synchronisation, was als *Overhead* bezeichnet wird. Bei einer Effizienz oberhalb von 1 liegt superlinearer Speedup vor [BBKS15]. Abbildung 4.1 zeigt die prinzipiellen Zusammenhänge von Speedup und Effizienz zur Anzahl der parallelen Slots (n_s).

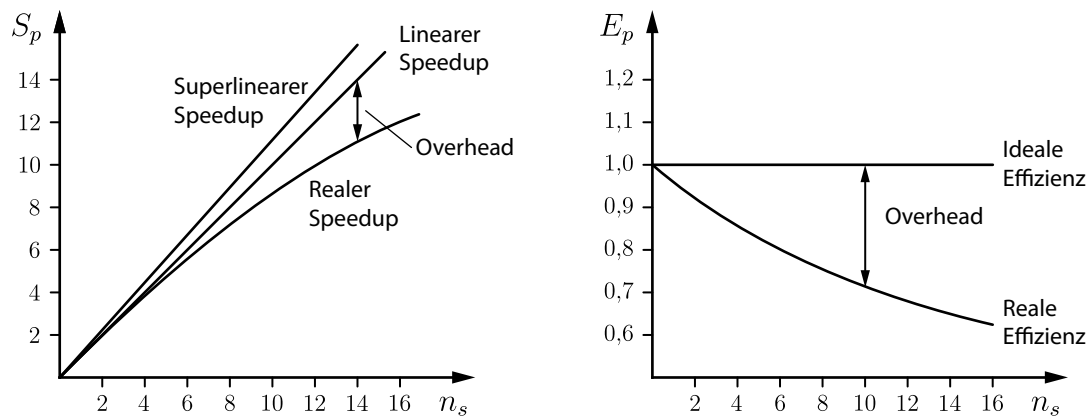


Abbildung 4.1: Verlauf von Speedup und Effizienz bei der parallelen Verarbeitung, nach [BBKS15]

Die Zerlegung eines zu parallelisierenden Problems wird als Dekomposition bezeichnet. Diese kann auf verschiedene Arten erfolgen: durch funktionale Zerlegung, durch Gebietszerlegung oder durch die Kombination beider Methoden.

Bei der funktionalen Zerlegung (engl. *function decomposition*) erfolgt die Zerlegung des zu bearbeitenden Problems aus funktionaler Sicht. Die Funktion zur Bearbeitung des Gesamtproblems wird dabei in voneinander unabhängige Teilfunktionen unterteilt, welche unabhängig voneinander parallel ausführbar sind. Somit liegt inhärenter Parallelismus vor. Existieren Datenabhängigkeiten zwischen den einzelnen Verarbeitungsschritten der Teilfunktionen, können diese durch eine statische Lastverteilung und eine statische Zuordnung der Teilfunktionen auf den Ressourcen berücksichtigt werden. Die Zuordnung der Teilfunktionen und Ressourcen wird als Allokation bezeichnet. Bei der statischen Allokation werden die Reihenfolge der Teilfunktionen und die verwendeten Ressourcen bereits im Vorfeld festgelegt [BBKS15]. Diese Vorgehensweise beruht auf der Annahme, dass die Zahl der Rechenoperationen, die benötigt wird, um eine Teilfunktion auszuführen, für alle Teilfunktionen gleich ist sowie alle Prozesse auf gleich schnellen Slots ausgeführt werden und somit eine homogene Umgebung vorherrscht. Ist eine dieser Annahmen nicht erfüllt, ergibt sich die Gesamtlaufzeit der parallelen Verarbeitung durch den langsamsten Slot bzw. die aufwendigste Teilfunktion [BM06]. Zudem ist die statische Allokation anfällig bei Ausfällen von Slots, da keine dynamische Neuverteilung der Teilfunktionen stattfindet.

Die Gebietszerlegung (engl. *domain decomposition*) geht von einer einmaligen Zerlegung der Eingabedaten des Gesamtproblems aus. Nach der Zerlegung werden die Eingabedaten auf mehrere parallele Slots verteilt und dort bearbeitet. Diese Verteilung erfolgt durch dynamische Allokation nach dem Master-Worker-Schema. Die Slots werden daher auch als Worker bezeichnet. Der prinzipielle Aufbau des Master-Worker-Schemas ist in Abbildung 4.2 dargestellt.

Die zerlegten Eingabedaten werden durch einen zentralen Master-Prozess auf n_s Worker verteilt, welche die Daten verarbeiten und die Ergebnisse zurück an den Master übermitteln. Das Master-Worker-Schema wird auch als Master-Slave-Modell bezeichnet und weist prinzipiell den gleichen strukturellen Aufbau wie ein Client-Server-System auf. Durch mehrfache rekursive Zerlegung der Daten können dabei auch ganze Berechnungsbäume abgebildet werden.

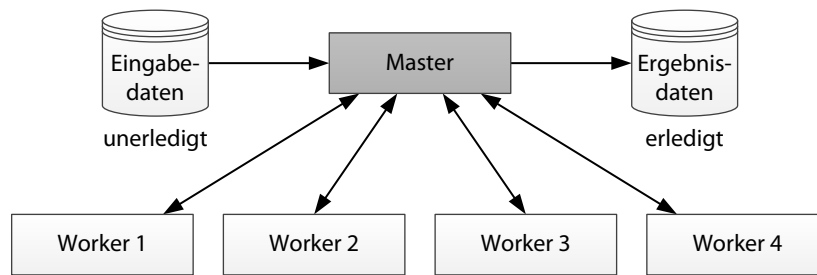


Abbildung 4.2: Prinzipieller Aufbau des Master-Worker-Schemas, nach [BM06], [BBKS15]

Der Master verwaltet dabei zwei Arten von Daten: die noch unerledigten und die bereits abgearbeiteten Daten. Bei der dynamischen Lastverteilung wird stets eine homogene Auslastung der Slots angestrebt. Schnelleren Slots wird automatisch mehr Arbeit zugewiesen als langsameren. Je feiner die Granularität der Datenzerlegung gewählt wird, desto mehr Kommunikation ist zwischen dem Master und den Workern notwendig. Obwohl die Größe der zu übertragenden Datenpakete abnimmt, erhöht sich die Arbeitslast des Masters. Bei sehr feiner Granularität kann dies sogar zu einer Überlastung des Masters führen und im Master entsteht ein Engpass des Gesamtsystems [BBKS15].

Für die Dekomposition und Parallelisierung von Optimierungsverfahren wird von den zwei grundsätzlichen Modellen einer Optimierung ausgegangen: dem Optimierungsmodell und dem Evaluationsmodell. Je nach Komplexität der Aufgabe kann innerhalb des Evaluationsmodells zusätzlich zwischen dem Entwurfsmodell und dem Analysemodell unterschieden werden (s. Abschnitt 2.1). Die Unterscheidung dieser Modelle liefert die Basis für die zwei grundlegenden Arten der Modelldekomposition: die Dekomposition des Optimierungsmodells und die des Evaluationsmodells.

Für die *Dekomposition des Optimierungsmodells* ist die Funktionsweise des Optimierungsverfahrens entscheidend. Es muss inhärenter Parallelismus vorliegen und die Evaluationen müssen unabhängig voneinander sein, zumindest in einer begrenzten Domäne, welche im Folgenden als Parallelisierungsdomäne bezeichnet wird. Eine Parallelisierungsdomäne kann z. B. durch die voneinander unabhängigen Individuen einer Generation eines GA (s. Abschnitt 2.6.2.1) oder durch die voneinander unabhängigen Stützstellen eines DoE-Versuchsplans (s. Abschnitt 2.7.1) gebildet werden.

Auch die voneinander unabhängigen Evaluationen einer numerischen Sensitivitätsanalyse stellen eine Parallelisierungsdomäne dar und können parallel durchgeführt werden. Hierbei werden die Einflüsse der einzelnen Designvariablen auf die Zielfunktion (Sensitivitäten) berechnet und anschließend daraus das Optimum ermittelt. Dies kann z. B. durch die Berechnung der Gradienten der Zielfunktion erfolgen. (s. Abschnitt 2.6.1). Beispielhaft hierzu ergeben sich bei der Berechnung der Sensitivitäten von 20 Designvariablen 21 unabhängige Evaluationen. Eine Evaluation bildet die Ausgangswerte der Designvariablen ab. Bei den restlichen Evaluationen wird jeweils eine Designvariable variiert und die Sensitivität dieser Designvariable bestimmt. Durch die parallele Bearbeitung der Evaluationen werden nur 5 % der sequentiellen Bearbeitungszeit benötigt. Das Evaluationsmodell bleibt dabei unverändert [Sch13].

Analog kann die parallele Evaluation einer Generation bei Verwendung eines Genetischen Algorithmus erfolgen. Alle Individuen einer Generation sind unabhängig voneinander und können parallel berechnet werden [SD08]. Eine Generation stellt dabei die

Parallelisierungsdomäne dar. Weitere Verfahren, welche die Dekomposition des Optimierungsmodells unterstützen, sind das Monte-Carlo-Verfahren, das Partikelschwarmverfahren (s. Abschnitt 2.6.2.2) und die explorative Untersuchung des Lösungsraumes mit Hilfe der statistischen Versuchsplanung (s. Abschnitt 2.7).

Bei der *Dekomposition des Evaluationsmodells* wird das Evaluationsmodell zerlegt und parallel verarbeitet. Die eigentliche Optimierung erfolgt sequentiell. Dabei kann die Dekomposition an verschiedenen Stellen des Evaluationsmodells stattfinden. So ist bei der expliziten FEM-Simulation, welche z. B. für Crashesimulationen genutzt wird, eine effiziente Parallelisierung durch Gebietszerlegung des FE-Modells in separate Bereiche möglich. Die Bereiche können sich dabei überlappen oder sich nur an den Grenzen berühren. Nach der Zerlegung des FE-Modells werden die Bereiche parallel berechnet, wobei ggf. zwischenzeitlich die Randbedingungen zwischen angrenzenden Bereichen ausgetauscht bzw. aktualisiert werden. Anschließend werden die Lösungen der einzelnen Bereiche zur Gesamtlösung zusammengefügt.

Eine weitere Möglichkeit der Dekomposition des Evaluationsmodells ist die parallele Berechnung unabhängiger Lastfälle bei multidisziplinären Optimierungsaufgaben. So lassen sich z. B. die Eigenfrequenzen und die statische Festigkeit eines Bauteils unabhängig voneinander berechnen. Die Teilergebnisse der Lastfälle laufen anschließend in der Berechnung des Zielfunktionswertes zusammen. Diese Form der Modelldekomposition wird oft bei multikriteriellen oder multidisziplinären Optimierungsaufgaben verwendet (s. Abschnitt 2.8.1 und 2.8.2). Hierbei kommen in der Regel teilweise und vollständig unabhängige Simulationen gemäß der bei der Entwicklung beteiligten Fachdisziplinen und -abteilungen zum Einsatz, in denen nur einzelne Designvariablen des Designvariablenvektors verwendet werden. Somit findet eine Dekomposition des Designvariablenvektors statt, wobei auch Überschneidungen der einzelnen Bereiche abgebildet werden können [BHSS92], [WRBB96], [BK97], [SK00].

Die Verarbeitung der einzelnen Modelle erfolgt bei der Dekomposition des Evaluationsmodells in der Regel nach dem Master-Worker-Prinzip. Das Evaluationsmodell wird durch den Master-Prozess zerlegt und an die zur Verfügung stehenden Worker-Prozesse übermittelt. Die Teillösungen laufen dann wieder im Master-Prozess zusammen.

In der Bearbeitung einer Optimierungsaufgabe in der Produktentwicklung ist die Evaluation der verschiedenen Produktvarianten in der Regel der rechenintensivste Prozess der Optimierung. Insbesondere der Einsatz numerischer Simulationen (z. B. FEM- oder CFD-Simulationen) führt zu sehr zeitintensiven Analysen, wodurch das Bestreben nach einer Effizienzsteigerung erhöht wird. Die Evaluation einer Produktvariante beinhaltet dabei meist die Modellgenerierung (Entwurfsmodell), das Lösen des numerischen Problems und die Ergebnisauswertung (Analysemodell). Auch diese Modelltrennung des Evaluationsmodells stellt eine Modelldekomposition dar, welche in den folgenden Betrachtungen in den Evaluationsprozessen *Modell generieren* und *Modell bewerten* resultiert. Die Modellgenerierung kann z. B. die Erstellung von CAD-Geometrie und das Generieren von Eingabedaten für numerische Simulation sein, die Modellbewertung z. B. eine FEM-Simulation dieses Modells. Die einfachste Form der Durchführung einer solchen Optimierung stellt die sequentielle Verarbeitung dar [WV15]. Wie in Abbildung 4.3 dargestellt, erfolgt jeder Prozessschritt sequentiell nacheinander.

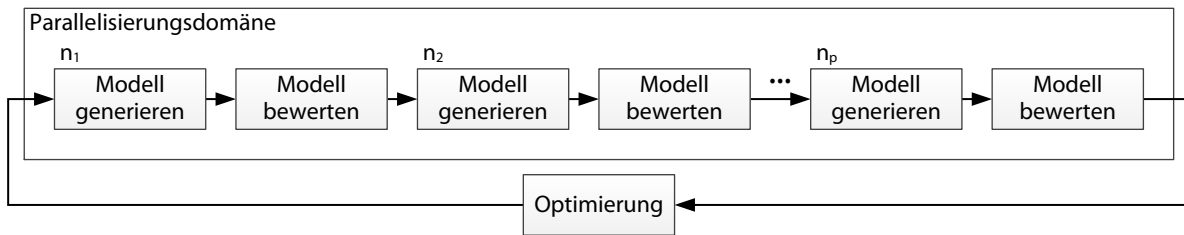


Abbildung 4.3: Sequentielle Verarbeitung von Optimierungsprozessen, nach [WV15]

Ausgehend von der rein sequentiellen Verarbeitung können durch die Modelldekomposition des Optimierungsmodells und des Evaluationsmodells zwei Arten der parallelen Verarbeitung durchgeführt werden: die vollständig parallele Verarbeitung und die parallele Simulationsverarbeitung.

Bei der vollständig parallelen Verarbeitung werden alle Produktvarianten parallel generiert und anschließend auch parallel bewertet. Bei der parallelen Simulationsverarbeitung erfolgt die Generierung der Produktvarianten sequentiell und die verschiedenen Lastfälle werden parallel verarbeitet [WV15]. Beide Arten der parallelen Verarbeitung sind in Abbildung 4.4 dargestellt.

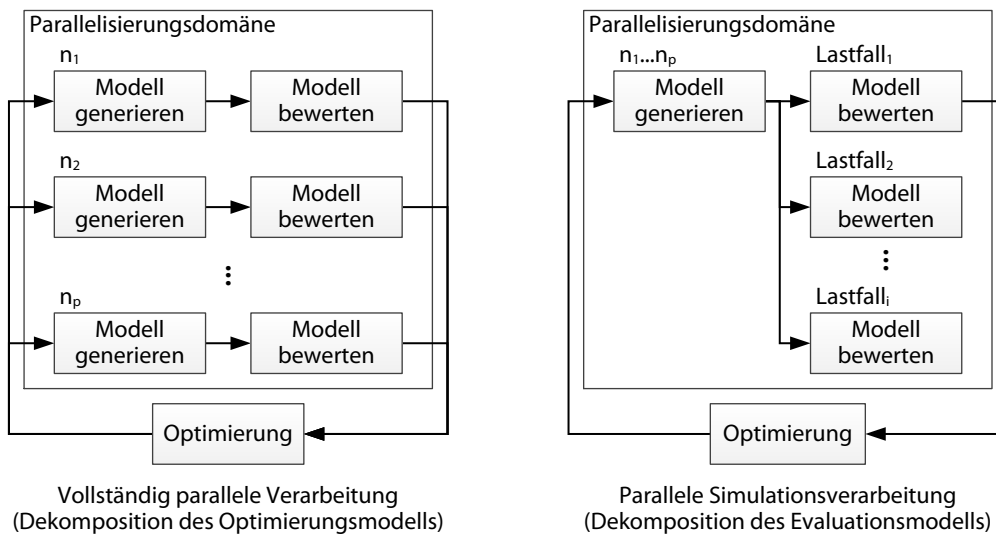


Abbildung 4.4: Parallele Verarbeitung von Optimierungsprozessen, nach [WV15]

Streng genommen wird bei der in Abbildung 4.4 dargestellten parallelen Simulationsverarbeitung nur das Analysemodell dekomponiert. Das Entwurfsmodell bleibt unverändert. Daran ist zu erkennen, dass die Dekomposition auch auf verschiedenen Stufen der verwendeten Modelle stattfinden kann.

Der Vorteil beider Parallelisierungsarten ist der geringere administrative Aufwand, insbesondere bei statischer Allokation und einem gemeinsam verwendeten Speicher. Die vollständig parallele Verarbeitung innerhalb einer Parallelisierungsdomäne ist sehr effizient, wenn die Anzahl der verwendeten Slots der Größe der Domäne oder zumindest einem ganzzahligen Teil dieser entspricht und alle verfügbaren Slots vollständig ausgenutzt werden. Dies führt jedoch auch zu einer großen Anzahl benötigter Ressourcen wie Hardware und Softwarelizenzen, insbesondere, wenn in kurzer Zeit große Domänen verarbeitet werden

sollen. Werden zur Evaluation der Produktvarianten kommerzielle Programme verwendet, was in der Regel der Fall ist, führt die limitierte Verfügbarkeit meist auch zu einer Limitierung der Parallelisierung [SSS11]. Zudem ist die vollständig parallele Verarbeitung sehr anfällig für spontan auftretende Engpässe. Fällt z. B. ein Slot aus oder eine benötigte Lizenz wird von einem externen Prozess verwendet, muss ein zweiter Zyklus erfolgen, um die letzte Produktvariante zu generieren und zu bewerten. Die übrigen Slots und Lizenzen werden in diesem Zyklus nicht verwendet und befinden sich im Leerlauf.

Die parallele Simulationsverarbeitung ist ebenfalls sehr einfach zu konfigurieren. Diese Methode ist zudem sehr nützlich bei der Verwendung deterministischer Optimierungsverfahren, da die Optimierung sequentiell ablaufen kann und nur die einzelnen Evaluationsprozesse parallel durchgeführt werden. Der Nachteil dieser Methode liegt in der Entstehung von Engpässen in der Simulation und Auswertung, insbesondere wenn viele verschiedene Lastfälle oder Simulationen mit unterschiedlichen Laufzeiten verwendet werden. In diesem Fall bestimmt der längste Evaluationsprozess die gesamte Verarbeitungszeit und die verfügbaren Slots verharren im Leerlauf.

Die beschriebenen Parallelisierungsmethoden berücksichtigen nicht die dynamische und effiziente Ausnutzung der verfügbaren Ressourcen, sei es der Hardware, der Software oder der Softwarelizenzen. In industriellen Anwendungen spielen jedoch diese Ressourcen eine wichtige Rolle. Aus diesem Grund sollte bei Parallelisierungsmethoden ein flexibles Ressourcenmanagement mit dem Ziel berücksichtigt werden, den Leerlauf der verfügbaren Ressourcen zu minimieren [WV15]. Dazu ist eine dynamische Allokation zwingend notwendig, bei der die Prozesselemente des Evaluationsprozesses den zur Verfügung stehenden Ressourcen dynamisch zugewiesen werden und somit auch Veränderungen in der Verfügbarkeit der Ressourcen berücksichtigt werden.

Beide in diesem Abschnitt beschriebenen Parallelisierungsmethoden von Optimierungsverfahren erfordern jeweils nur eine Dekomposition des relevanten Modells, das andere Modell bleibt unverändert. So wird bei der Dekomposition des Optimierungsmodells auch nur dieses Modell zerlegt, das Evaluationsmodell wird nicht verändert. Analog wird bei der Dekomposition des Evaluationsmodells verfahren. Hierbei bleibt das Optimierungsmodell unverändert. Da bei beiden Arten der Modelldekomposition das jeweils andere Modell stets unverändert bleibt, ist auch die Kombination der beiden Parallelisierungsmethoden möglich. So kann z. B. die parallele Evaluation der Individuen einer Generation bei Verwendung eines GA problemlos mit der parallelen Berechnung verschiedener Lastfälle der jeweiligen Produktvarianten erfolgen, was den Parallelisierungsgrad erhöht und zu zusätzlichen Freiheiten bei der dynamischen Allokation führt.

4.2 Anwendung von Parallelisierungsmethoden in der Optimierung

Zur Dekomposition des Optimierungsmodells und der parallelen Verarbeitung eines Optimierungsverfahrens müssen die Evaluationen mindestens in einer begrenzten Domäne unabhängig voneinander sein. Diese Domäne wird als Parallelisierungsdomäne bezeichnet (s. Abschnitt 4.1). Neben der parallelen Durchführung der numerischen Sensitivitätsanalyse deterministischer Optimierungsverfahren eignen sich zur Parallelisierung vor allem stochastische Verfahren (z. B. Genetische Algorithmen, Partikelschwärme), da diese eine

hohe Wahrscheinlichkeit liefern, das globale Optimum der Optimierungsaufgabe zu finden und durch die Parallelisierung der Nachteil der hohen Anzahl an benötigten Evaluationen reduziert wird.

Zudem arbeiten stochastische Optimierungsverfahren mit voneinander unabhängigen Evaluationen, welche direkt zur parallelen Verarbeitung verwendet werden können. Ein sehr häufig eingesetztes stochastisches Verfahren sind Genetische Algorithmen (GA). Diese arbeiten bereits von ihrem Grundgedanken aus parallel, da die Individuen generationsweise ausgewertet werden. Die Individuen einer Generation werden dabei unabhängig voneinander erstellt und evaluiert. Innerhalb einer Generation weisen GA somit inhärenten Parallelismus auf (s. Abschnitt 4.1).

Zur Effizienzsteigerung eines GA durch parallele Verarbeitung existieren drei grundsätzliche Methoden, welche sich nach dem verwendeten Populationsmodell unterscheiden: die globale Verteilung einer Population nach dem Master-Worker-Prinzip (globales Modell), die grobgranulare Verteilung mehrerer Populationen (regionales Modell) und die feingranulare Verteilung einer Population (lokales Modell) [CP98]. Die prinzipielle Struktur dieser Methoden ist in Abbildung 4.5 dargestellt. Die jeweiligen Populationsmodelle werden durch die hellgrauen Bereiche repräsentiert. Die dunkelgrauen Kreise stellen die Individuen dar. Zur Erhöhung der Übersichtlichkeit ist bei der feingranularen Verteilung nur das lokale Umfeld eines Individuums abgebildet.

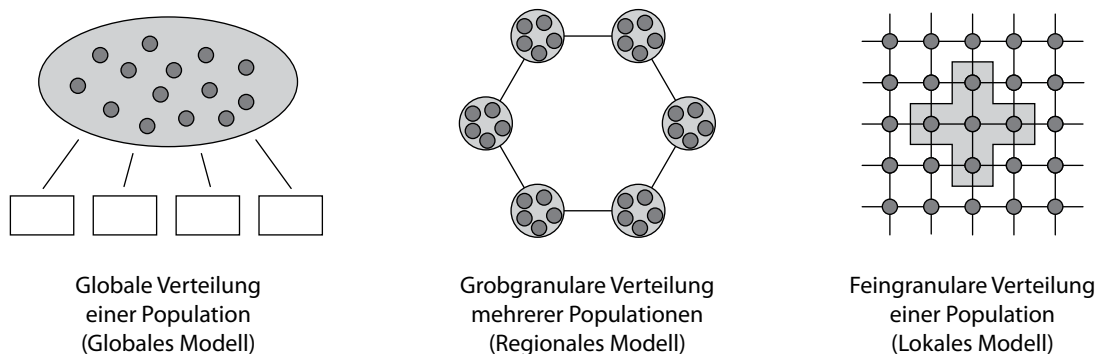


Abbildung 4.5: Parallelisierung genetischer Algorithmen, nach [CP98], [Poh00], [AT02]

Die einfachste Art der Parallelisierung eines GA ist die *globale Verteilung einer Population nach dem Master-Worker-Prinzip* (s. Abbildung 4.2). Der Optimierungsprozess läuft als Master-Prozess und steuert den Aufruf der Evaluationen der einzelnen Individuen. Die Evaluationen sowie die Berechnung der Zustandsvariablen und Zielfunktionswerte der Individuen erfolgen in den Worker-Prozessen. Der Master-Prozess verwendet die berechneten Zielfunktionswerte zur Ermittlung der Fitnesswerte und erzeugt die nächste Generation. Der Grad der Parallelisierung wird dabei über die Populationsgröße bestimmt. Stehen mehr Ressourcen als Individuen zur Verfügung, muss die Populationsgröße und somit der Parallelisierungsgrad erhöht werden [SG15]. Die globale Verteilung einer Population ist nicht an eine bestimmte Hardwarearchitektur gebunden, sondern kann auf verschiedenen Systemen implementiert werden [CP98].

Fritzsche et al. [FKBV12] beschreiben zwei Ansätze der Implementierung: die Parallelisierung der einzelnen Evaluationen der Individuen einer Generation und die Parallelisierung mit einer verteilten virtuellen Maschine. Bei dem ersten Ansatz wird jedes Individuum

einer Generation, welches nur durch ein Genom bzw. einen Parametersatz definiert wird, vollständig auf einen Worker-Rechner übertragen. Dieser führt die gesamte Evaluation durch und berechnet den Zielfunktionswert des Individuums. Der Zielfunktionswert wird dann zurück an den Master-Rechner übertragen. Diese Methode funktioniert nur in einem homogenen Cluster wirklich effizient. Um gleiche Evaluationszeiten zu gewährleisten, müssen alle verwendeten Rechner die benötigte Software und gleichwertige Hardware besitzen. In der Weiterentwicklung dieser Methode wird die Evaluation der Individuen in Teilprozesse zerlegt, welche dann parallel verarbeitet werden. Dadurch wird gerade bei komplexen Evaluationen die Flexibilität erhöht. Jedoch beinhaltet die Methode kein flexibles Ressourcenmanagement. Die verwendete Optimierungssoftware muss unter einem angemeldeten Benutzerkonto auf jedem Rechner ausgeführt werden. Dies schließt die spontane Unterbrechung einer laufenden Optimierung auf einem Rechner zur interaktiven Nutzung dieses Rechners durch einen anderen Anwender aus. Zudem können Rechner, welche nicht interaktiv verwendet werden, nur umständlich dem Cluster hinzugefügt werden. Die für die Optimierung genutzte Cluster-Umgebung kann somit nicht dynamisch auf die veränderliche Verfügbarkeit der Ressourcen reagieren.

Der zweite Ansatz sieht die Verwendung einer verteilten virtuellen Umgebung vor, welche aus einzelnen Rechnern aufgebaut ist. Durch das Zusammenfügen der Hardwareressourcen zu einem virtuellen Rechner steht für die Optimierung ein hohes Maß an Hardwareressourcen zur Verfügung (CPU und Speicher). Die Evaluationsprozesse können innerhalb des virtuellen Rechners sehr flexibel verteilt und parallel verarbeitet werden. Der Nachteil dieses Ansatzes ist der hohe administrative Aufwand, um den Optimierungsprozess und die für die Evaluation benötigte Software innerhalb der virtuellen Umgebung stabil zu verwenden [FKBV12].

Die *Grobgranulare Verteilung mehrerer Populationen* basiert auf einzelnen Subpopulationen, welche den jeweiligen Slots zugeordnet sind, auf denen die Evaluationsprozesse verarbeitet werden. Entwickeln sich die Populationen unterschiedlich, werden die besten Individuen zwischen den Subpopulationen in regelmäßigen Abständen ausgetauscht. Dieser Prozess wird als Migration bezeichnet. Da die Subpopulationen sich analog zu Lebewesen auf Inseln unabhängig von einander entwickeln und nur definierte Migration stattfindet, wird dieser Aufbau auch als Insel-Migrationsmodell bezeichnet [RN04]. Durch die Verwendung von Subpopulationen und die dadurch resultierenden zusätzlichen Parameter wie Anzahl und Größe der Subpopulationen, Migrationsrate und Inselstruktur, wird die Komplexität der Anwendung dieser Algorithmen erhöht [CP98]. Da hier die Zuordnung der Subpopulationen in der Regel durch statische Allokation erfolgt, ist diese Methode nicht sehr flexibel einsetzbar und es ergeben sich die gleichen Nachteile wie bei der feingranularen Verteilung.

Parallele GA mit *feingranularer Verteilung einer Population* verwenden nur eine Population, welche durch eine räumliche Struktur segmentiert ist, wodurch die Interaktion zwischen den Individuen eingeschränkt wird. Jedes Individuum kann nur mit seinen direkten Nachbarn konkurrieren und interagieren (s. grauer Bereich in Abbildung 4.5). Da jedes Individuum mehrere Nachbarn hat und sich die Bereiche benachbarter Individuen überlappen, können sich gute Lösungen schrittweise über die gesamte Population ausbreiten. Ein Beispiel für diese Parallelisierungsmethode ist der parallele GA nach Mühlenbein [Müh91], [Müh92]. Hierbei sucht jedes Individuum durch lokales Hillclimbing nach dem Optimum in seiner Nähe. Anschließend erfolgt die Erzeugung neuer Individuen durch Crossover mit einem Partnerindividuum aus dessen Nachbarschaft. Dieser Ablauf wird

solange wiederholt, bis das Abbruchkriterium erfüllt ist [GKK04]. Im Idealfall verfügt bei dieser Methode jedes Individuum über einen Slot, auf dem die Evaluation durchgeführt wird. Die Populationsgröße bestimmt also die verwendeten Slots oder umgekehrt. Ein flexibles Ressourcenmanagement mit dynamischer Allokation ist somit nicht direkt möglich. Weiterhin muss auf jedem Slot die identische Software vorhanden sein und es müssen ausreichend Lizenzen zur Verfügung stehen. Zudem ist die Methode sehr anfällig für Ausfälle einzelner Slots, wenn eine feste Populationsgröße verwendet wird. Durch eine variable Populationsgröße kann die Flexibilität dieser Methode erhöht werden, wodurch sich jedoch die Effektivität des GA reduziert.

Durch die Kombination der beschriebenen Methoden zur Parallelisierung von GA sind zudem zahlreiche Sonderformen hierarchisch bzw. hybrid paralleler GA entstanden, welche meist speziell auf die jeweilige Problemstellung angepasst sind. Eine Übersicht verschiedener Sonderformen ist in [CP98], [AT02] zu finden.

Die parallele Bearbeitung von Partikelschwärmen erfolgt grundsätzlich analog zu den beschriebenen Parallelisierungsmethoden genetischer Algorithmen. Basierend auf dem Insel-Modell mit grobgranularer Verteilung lässt sich der Partikelschwarm in Subschwärme zerlegen, welche verschiedene Bereiche des Designvariablenraumes abdecken. Bei der multikriteriellen Optimierung werden somit durch die einzelnen Schwärme unterschiedliche Bereiche der Pareto-Front berechnet. Ist die Kommunikationsfähigkeit zwischen den Subschwärmen limitiert, kann bei diesem Vorgehen nicht sichergestellt werden, dass ausschließlich separate Bereiche der Pareto-Front bearbeitet werden und keine Überlappungen entstehen. Daher übernimmt ein den Subschwärmen übergeordneter Master-Prozess die Ergebnisverwaltung der Subschwärme und ggf. eine Neuordnung der jeweiligen Bereiche. Zudem wird durch den Master-Prozess die Lastverteilung bei heterogenen Ressourcen über Wartezeiten der Subpopulationen gesteuert [Mos10].

Auch hybride Optimierungsverfahren, welche analog zu stochastischen Verfahren teilweise oder vollständig voneinander unabhängige Evaluationen verwenden, können parallelisiert werden (s. Abschnitt 2.6.3). Dazu zählen Kombinationen aus Optimierungsverfahren mit einem stochastischen Anteil oder Verfahren zur explorativen Untersuchung des Lösungsraumes mittels statistischer Versuchsplanung [RB13]. Exemplarisch ist die Optimierung auf Basis parallel generierter Metamodelle zu nennen, wobei der DoE-Versuchsplan auf heterogenen Workstations in einem Cluster bearbeitet wird [TV02].

Ein alternativer Ansatz zur Parallelisierung von Optimierungsverfahren ist die parallele Verwendung verschiedener Optimierungsalgorithmen. In dieser Multi-Algorithmus-Infrastruktur konkurrieren verschiedene Algorithmen parallel miteinander, wodurch die Wahrscheinlichkeit erhöht wird, das globale Optimum zu finden. Dabei wird ein globales Abbruchkriterium verwendet, in das die jeweiligen Lösungen der Algorithmen einfließen. Diese Parallelisierungsmethode ist sehr rechenintensiv und mit einer hohen Anzahl verfügbarer Ressourcen verbunden. In Untersuchungen mit dieser Methode werden große homogene Cluster von bis zu 128 Workstations verwendet. Zudem muss eine große Anzahl von Softwarelizenzen vorhanden sein [GH02].

Die beschriebenen Parallelisierungsmethoden zeigen die große Bandbreite der Verwendung von Parallelverarbeitung zur Steigerung der Effizienz von Optimierungsverfahren oder der explorativen Untersuchung des Lösungsraumes. Dabei werden jedoch nur selten die verwendeten Ressourcen und eine Limitierung dieser Ressourcen berücksichtigt. Es werden lediglich die benötigten Slots zur Parallelisierung betrachtet. Installierte Software und zur

Verfügung stehende Softwarelizenzen finden keine Berücksichtigung. Insbesondere wenn zur Evaluation der Produktvarianten kommerzielle Software, z. B. FE-Solver, eingesetzt werden, können dieses den Grad der Parallelisierung beschränken [SSS11].

Weiterhin stellt das Master-Worker-Prinzip die flexibelste und universell einsetzbare Methode zur Parallelisierung dar, bei der keine weitere funktionale Dekomposition des Algorithmus erfolgen muss und sich somit eine Vielzahl von Algorithmen für die Anwendung eignen. Über dynamische Allokation kann nach dem Master-Worker-Prinzip ein flexibles Ressourcenmanagement unterschiedlicher Arten von Ressourcen realisiert werden, bei dem auch spontan auf Ausfälle oder zusätzlich verfügbare Ressourcen reagiert werden kann.

4.3 Concurrent und Simultaneous Optimization als Ansatz zur Effizienzsteigerung

Bei der Beschreibung der Anwendung von Parallelisierungsmethoden in der Optimierung wird deutlich, dass die Effizienz und die Parallelisierung von Optimierungsverfahren durch drei wesentliche Faktoren bestimmt werden: die Effektivität, die verwendeten Ressourcen und die Laufzeit der Optimierung. Die Definitionen der Begriffe Effektivität und Effizienz von Optimierungen finden sich in Abschnitt 2.1.

Analog zum Magischen Dreieck des Projektmanagements, welches die gegenseitigen Abhängigkeiten der Projektziele bzw. der Qualität, der Kosten bzw. der Ressourcen und der Projektdauer bzw. der Zeit gegenüberstellt [SBL98], [KHL⁺11], können durch diese Faktoren die Ziele einer Optimierung abgebildet werden. Erfolgt keine grundsätzliche Veränderung der Optimierungsstrategie (s. Abschnitt 2.8), z. B. durch einen Austausch des Optimierungsalgorithmus, kann keiner dieser Faktoren verändert werden, ohne dass einer der anderen Faktoren beeinflusst wird. Der Zusammenhang der drei Faktoren einer Optimierung ist in Abbildung 4.6 dargestellt.

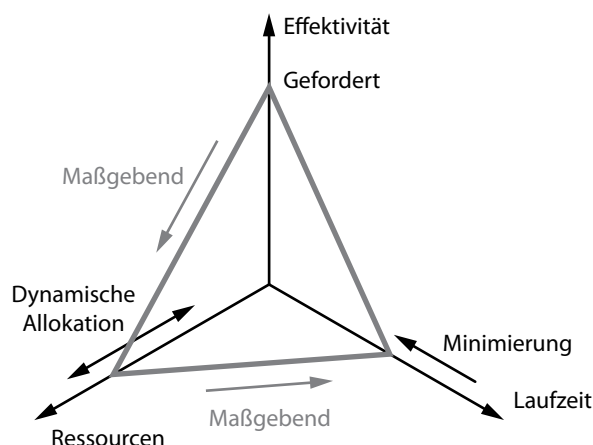


Abbildung 4.6: Zusammenhang von Effektivität, Ressourcen und Laufzeit einer Optimierung

Soll z. B. die Effektivität einer Optimierung erhöht werden, weil nicht das globale Optimum gefunden wurde, wird dazu in der Regel eine größere Anzahl an Evaluationen benötigt, wodurch sich bei einer gleichen Anzahl an Ressourcen die Laufzeit der Optimierung

erhöht. Werden bei der Optimierung mehr Ressourcen verwendet, kann stärker parallelisiert werden und somit die Laufzeit wieder reduziert werden. Das Ziel bei der Bestimmung der Faktoren sollte immer von einer durch den Anwender geforderten Effektivität ausgehen, wodurch sich die Parameter der Optimierung ergeben, z. B. die Populationsgröße eines GA. Durch dynamische Allokation der Evaluationsprozesse sollten immer die maximal zur Verfügung stehenden Ressourcen mit dem Ziel verwendet werden, die Laufzeit der Optimierung zu minimieren und somit die Effizienz der Optimierung zu steigern.

Unter den Begriffen Concurrent Optimization und Simultaneous Optimization wird im Folgenden ein sehr flexibler und effizienter Ansatz zur Parallelisierung von Optimierungsverfahren vorgestellt. Dieser Ansatz basiert auf einer Analogiebetrachtung von Produktentwicklungsprozessen und Optimierungsverfahren, welche beide einen iterativen Prozess gemäß des TOTE-Schemas darstellen. Ausgehend von einer Beschreibung der Methoden zur Parallelisierung von Produktentwicklungsprozessen werden wesentliche Charakteristiken dieser Methoden zusammengefasst und auf die Verwendung in Optimierungsverfahren übertragen.

Die einfachste und natürlichste Form der Bearbeitung aufeinanderfolgender Aufgaben ist die sequentielle Bearbeitung. Da die Aufgaben bei dieser Form der Bearbeitung strikt nacheinander bearbeitet werden, ist nur ein geringer Aufwand an Organisation und Planung erforderlich. Das ursprünglich zur Beschreibung von Entwicklungsprozessen in der Softwareentwicklung erstellte Wasserfall-Modell beschreibt diese sequentielle Form der Bearbeitung [Roy70], [BT76]. Die einzelnen Phasen sind dabei als voneinander abhängige Stufen dargestellt und eine Phase muss erst abgeschlossen sein, bevor die darauf folgende Phase beginnen kann. Das Wasserfall-Modell stellt eine sehr generalisierte Abbildung der Aktivitäten dar und gibt dem Anwender dieses Modells lediglich eine Orientierung über die sequentielle Abfolge der Tätigkeiten und Prozesse während der Entwicklung. Die Schnittstellen zwischen den einzelnen Aktivitäten bilden Meilensteine und Präsentationen von Ergebnissen. Aufgrund der generalisierten Sichtweise dieses Modells ist die Anwendung vor allem für Projektmanager geeignet, um schnell den Projektfortschritt zu überblicken und die verbleibende Zeit bis zur Fertigstellung abzuschätzen [Pfl98].

Concurrent Engineering (CE) und Simultaneous Engineering (SE) hingegen beschreiben eine Philosophie zur Parallelisierung von Aktivitäten in Produktentwicklungsprozessen mit dem Ziel, die Produktentwicklung in der Praxis zu verbessern, welche bereits erfolgreiche Anwendung in der Industrie findet [WPBS88], [BHK11]. Durch die gleichzeitige Berücksichtigung von Funktionalität, Zuverlässigkeit, Herstellbarkeit, und Marktfähigkeit bei der Entwicklung von neuen Produkten sollen die Produktentwicklungszeit verkürzt, Produktlebenszykluskosten verringert, die Wertigkeit des Produktes gesteigert und eine höhere Produktqualität sichergestellt werden [WPBS88], [NW89], [MAAE+95].

Winner et al. [WPBS88] definieren CE als einen systematischen Ansatz für die integrierte und parallele Entwicklung von Produkten und die damit verbundenen Prozesse, einschließlich der Herstellung und des Produktsupports. Mit Hilfe dieses Ansatzes sollen Produktentwickler von Beginn an alle Elemente des Produktlebenszyklus von der Konzeption bis zur Entsorgung berücksichtigen, einschließlich der Qualität, der Kosten, des Zeitplans und der Kundenanforderungen [WPBS88]. Ausgehend von dieser Definition haben sich weitere Definitionen etabliert.

So definiert Cleetus [Cle92] CE auch als systematischen Ansatz, in dem jedoch verstärkt die Erwartungen des Kunden hervorgehoben werden und bei der sehr stark parallelisierten

Arbeit des Projektteams gemäß der verschiedenen Perspektiven des Produktlebenszyklus nur kurze Abstimmungen zum Abgleichen der Ergebnisse stattfinden [Cle92].

Pawar et al. definieren CE als eine strukturierte Form der Organisation und Verwaltung der Entwicklung von Produkten oder Dienstleistungen unter Berücksichtigung der Integration von Ressourcen und Zeitplänen, der Aufteilung gemeinsamer Ziele und der Bereitstellung präziser Informationen [PDT⁺96].

In der Definition durch Romero et al. [RDE⁺15] stellt CE ebenfalls eine Philosophie des Arbeitens dar, basierend auf der gleichzeitigen Verfügbarkeit und Bearbeitung der lebenszyklusrelevanten Informationen während der Entwicklung eines Produktes. Die Umsetzung dieser Philosophie kann dabei auf Ebene der Produktentwicklung, der Arbeitsplanung oder der Kommunikation erfolgen [RDE⁺15].

In den vorgestellten Definitionen findet keine Trennung zwischen CE und SE statt. In erster Linie wird der Begriff CE verwendet oder es finden beide Begriffe analog Anwendung [WPBS88], [MAAE⁺95]. Da die parallele Bearbeitung in der Produktentwicklung ebenfalls auf unterschiedliche Arten erfolgen kann, wird für die weiteren Betrachtungen in dieser Arbeit zwischen CE und SE unterschieden.

Beim CE wird eine komplexe Aufgabe auf mehrere Personen eines Teams aufgeteilt. Die Teammitglieder arbeiten dann parallel an ihrem spezifischen Teil dieser Aufgabe. Für die Durchführung von CE ist die Definition physisch und logisch abgegrenzter Teilaktivitäten mit eindeutigen Schnittstellen notwendig, z. B. durch die Festlegung von Bauraumschnittstellen bei der CAD-Modellierung. Die Ergebnisse der jeweiligen Teilaktivitäten werden nach der Fertigstellung wieder zusammengeführt und miteinander abgeglichen.

Im SE können sich aufeinanderfolgende Aktivitäten überlappen und parallel durchgeführt werden, um effizient hinsichtlich Kosten und Zeit zu arbeiten. So können z. B. die Konstruktion eines Bauteils und die Planung der Fertigungsprozesse parallel durchgeführt werden. Wie in Abbildung 4.7 dargestellt, kann die parallele Bearbeitung von Aktivitäten in der Produktentwicklung durch CE, SE und durch ihre Kombination erfolgen. Durch diese parallele Bearbeitung der Aktivitäten wird zwar eine Verkürzung der Gesamtbearbeitungszeit erreicht, es werden jedoch auch zusätzliche Ressourcen benötigt, was die in Abbildung 4.6 dargestellten Abhängigkeiten widerspiegelt.

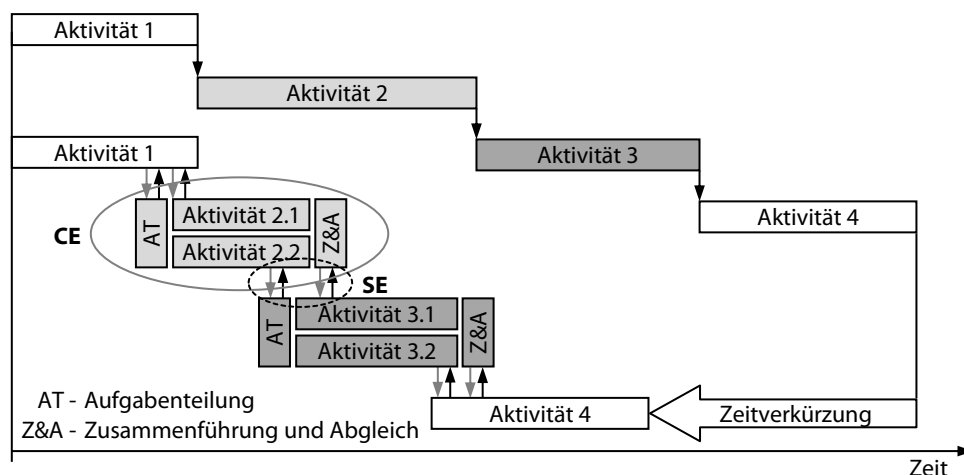


Abbildung 4.7: Concurrent Engineering und Simultaneous Engineering [Vaj14]

Die wichtigsten Kriterien für die Parallelisierung von Prozessen und Aktivitäten durch CE und SE sind die Verfügbarkeit und die Stabilität der benötigten Informationen für eine Aktivität, insbesondere die Ergebnisse der vorher begonnenen Aktivitäten. Die Ergebnisse sollten soweit stabil sein, dass die statistische Wahrscheinlichkeit einer Änderung und die damit verbundenen Änderungskosten geringer sind als die Kosten, die durch zu spätes Weiterarbeiten entstehen würden [VWBZ09].

Seit der Einführung von CE und SE haben sich viele Arbeiten mit dem Einsatz computerunterstützter Systeme zur Unterstützung von SE beschäftigt, wobei sich diese jedoch vor allem auf die damit verbundenen kollaborativen Prozesse fokussieren und nicht die Parallelisierung der Aktivitäten selbst [MAAE⁺95].

Obwohl sich die beschriebenen Definitionen des CE und SE unterscheiden, weisen sie doch einige wesentliche gemeinsame sowie ergänzende Charakteristiken auf. So gehen z. B. die meisten Definitionen von einem strukturierten systematischen Ansatz aus und beinhalten die parallele Bearbeitung von Aufgaben in einem integrierten Produktentwicklungsumfeld, was in den meisten Fällen auch die Arbeit in interdisziplinären Teams bedingt. Folgende Charakteristiken werden im CE und SE verwendet:

- Strukturierter systematischer Ansatz
- Integriertes Produktentwicklungsumfeld
- Einsatz interdisziplinärer Teams
- Erfüllung gemeinsamer Ziele
- Effiziente Nutzung vorhandener Ressourcen
- Parallele Bearbeitung von Aktivitäten
- Verfügbarkeit und Stabilität benötigter Informationen

Ausgehend von diesen Charakteristiken können die Methoden des CE und des SE auf die Verwendung in Optimierungsverfahren übertragen werden, da die Aktivitäten und Prozesse in der Produktentwicklung auch als kontinuierlicher Optimierungsprozess, basierend auf dem TOTE-Schema (s. Abschnitt 2.3), gesehen werden können und somit eine gewisse Konformität zwischen Produktentwicklungs- und Optimierungsprozessen besteht, wie auch in den beschriebenen Analogiebetrachtungen der biologisch inspirierten Programmierung und der Autogenetischen Konstruktionstheorie herausgestellt wurde (s. Kapitel 3).

Der Transfer der Methoden des CE und des SE auf die Verwendung in Optimierungsverfahren wird durch die Begriffe *Concurrent Optimization* und *Simultaneous Optimization* beschrieben. Analog zur Parallelisierung von Produktentwicklungsprozessen werden bei *Concurrent Optimization* (CO) komplexe Evaluationsprozesse auf mehrere Slots eines Clusters aufgeteilt. Dieser Ansatz repräsentiert somit eine vollständig parallele Verarbeitung bzw. eine parallele Simulationsverarbeitung, welche die bereits bekannten konventionellen Formen der Parallelisierung darstellen (s. Abbildung 4.4).

Bei *Simultaneous Optimization* (SO) können sich einzelne Teile der Evaluationsprozesse überlappen und parallel verarbeitet werden. Das Ziel dieses Ansatzes ist es, ausgehend von einer Dekomposition des Optimierungs- sowie des Evaluationsmodells (s. Abschnitt 4.1), die Evaluationsprozesse in Bezug auf die maximal zur Verfügung stehenden Ressourcen

(z. B. Rechner, Software, Lizenzen) zu parallelisieren und somit den Leerlauf der verfügbaren Ressourcen zu minimieren und die Ressourcen möglichst effizient zu nutzen [WJV15], [WV15].

Durch die Dekomposition des Evaluationsmodells entstehen mehrere zusammenhängende und voneinander abhängige Prozesselemente, welche den Evaluationsprozess beschreiben. Diese Prozesselemente bilden einen Workflow, in dem aufgrund der Abhängigkeiten der Prozesselemente die Reihenfolge der Bearbeitung festgelegt ist. Ein Workflow repräsentiert eine fest verkettete Folge von Arbeitsschritten bzw. Prozessen oder Prozesselementen, welche nicht oder nur sehr selten geändert werden [VWBZ09]. Haben dabei z. B. zwei Prozesselemente ein gemeinsames Vorgängerelement, können diese bei ausreichend zur Verfügung stehenden Ressourcen auch parallel bearbeitet werden.

Durch die Dekomposition des Optimierungsmodells (z. B. Individuen einer Generation eines GA) entstehen weitere Evaluationsprozesse, welche inhärenten Parallelismus aufweisen (s. Abschnitt 4.1) und ebenfalls in parallelisierbare Prozesselemente dekomponiert werden können.

Diese vollständige Zerlegung beider Modelle resultiert in einer Vielzahl unterschiedlicher Prozesselemente, welche durch verschiedene Abhängigkeiten gekennzeichnet sind. Maßgebend für die Abhängigkeiten sind dabei die zur Bearbeitung der Prozesselemente benötigten Informationen bzw. Eingabedaten, welche in der Regel durch die Vorgängerelemente des Workflows bereitgestellt werden. Weiterhin verfügen die Prozesselemente aufgrund der Multidisziplinarität einer Optimierungsaufgabe (s. Abschnitt 2.8.2) über verschiedene Anforderungen an die zur Bearbeitung benötigten Ressourcen.

Analog zu CE und SE basieren CO und SO bei der Parallelisierung nur auf der Verfügbarkeit und der Stabilität der für ein Prozesselement benötigten Informationen sowie der notwendigen Ressourcen. Somit kann eine große Flexibilität in der parallelen Bearbeitung der Prozesselemente gewährleistet und dynamisch auf Engpässe bei den Ressourcen reagiert werden. Liegen die benötigten Informationen vor, z. B. durch die Fertigstellung eines vorgelagerten Prozesselementes und steht eine Ressource zur Verfügung, wird das Prozesselement bearbeitet. Auf diese Art können verschiedene Prozesselemente integriert und eine multidisziplinäre Optimierung oder DoE-Studie effizient durchgeführt werden.

Ein Anwendungsbeispiel von CO und SO in kombinierter Form ist in Abbildung 4.8 dargestellt. Das Evaluationsmodell besteht aus einem Entwurfsmodell und einem Analysemodell, welche in den Prozesselementen *Modell generieren* und *Modell bewerten* resultieren. Der Prozess der Modellbewertung kann dabei mehrere Lastfälle beinhalten, welche nach dem CO bzw. der parallelen Simulationsverarbeitung parallelisiert werden. Zusätzlich ist die Ressource zur Modellgenerierung limitiert (z. B. aufgrund begrenzter Lizenzen), was zu einem Engpass in der Bearbeitung führt. Die Bewertung der Modelle ist nicht beschränkt und kann in vollem Maß parallel durchgeführt werden.

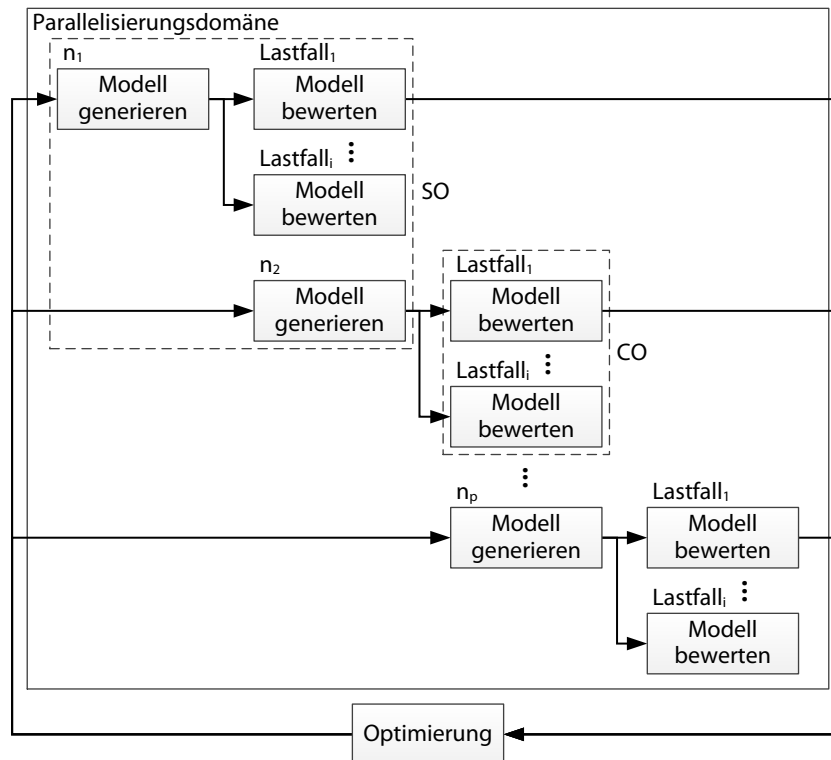


Abbildung 4.8: Grundlegender Ablauf der Concurrent und Simultaneous Optimization-Methode

Aufgrund der Restriktion in der Modellgenerierung muss die Optimierung nach konventionellen Methoden sequentiell oder rein durch CO bzw. parallele Simulationsverarbeitung erfolgen, was nicht effizient ist, da sich in dieser Zeit die Ressource zur Modellgenerierung im Leerlauf befindet. Die Optimierung kann deutlich schneller als SO durchgeführt werden. Hierbei wird ein neues Modell generiert, wenn die Generierung des Vorgängermodells abgeschlossen ist und die vorhandene Lizenz wieder zur Verfügung steht. Sobald die benötigte Ressource zur Modellgenerierung verfügbar ist und die erforderlichen Informationen (z. B. Eingabedaten) vorhanden sind, wird der Prozess ausgeführt [WV15].

Die Slots der verteilten IT-Umgebung verhalten sich dabei analog zu den Mitgliedern eines interdisziplinären Entwicklungsteams, welche unterschiedliche Qualifikationen besitzen und somit unterschiedliche Tätigkeiten ausführen können. Fällt spontan ein Mitglied des Teams aus oder es kommt ein neues dazu, erfolgt eine Neuverteilung der Aufgaben durch dynamische Allokation mit dem Ziel, die Gesamtaufgabe möglichst effizient zu bearbeiten und die Bearbeitungszeit zu minimieren. Die Dekomposition des Evaluationsmodells sollte dazu gemäß der verwendeten Ressourcen, z. B. gemäß der benötigten Software oder Hardware erfolgen.

4.4 Verteiltes Rechnen

Die in Abschnitt 4.3 vorgestellte Methode der Concurrent und Simultaneous Optimization zur effizienten Parallelisierung von Evaluationsprozessen erfordert eine verteilte IT-Umgebung, um die parallele Bearbeitung der Evaluationsprozesse zu ermöglichen. Zur

parallelen und effizienten Bearbeitung von rechenintensiven Problemen in der Wissenschaft sowie in der Produktentwicklung ist das verteilte Rechnen unabdingbar. Hierbei haben sich die folgenden grundlegenden Herangehensweisen etabliert:

- High-Performance Computing (HPC): Das Ziel des HPC ist das Durchführen eines Berechnungsjobs in möglichst kurzer Zeit. Daher werden Supercomputer, welche nach dem HPC-Prinzip arbeiten, gemäß der Anzahl der Gleitkommaoperationen pro Sekunde (FLOPS) bewertet [HDF12].
- High-Throughput Computing (HTC): Beim HTC steht das Lösen einer großen Anzahl an Berechnungsjobs im Fokus, wie es auch bei der Parallelisierung von Optimierungsaufgaben der Fall ist. Dabei soll eine möglichst große Rechenkapazität über einen längeren Zeitraum bereitgestellt werden. Hier zählen vorrangig die Anzahl der Berechnungen pro Tag, Monat oder Jahr [Cen15].

Zur Realisierung der verteilten IT-Umgebung, welche die Grundlage zur Umsetzung der entwickelten Parallelisierungsmethode bildet, wird das Cluster-Computing verfolgt. Auf diese Weise lassen sich freie Rechenkapazitäten vorhandener Rechner innerhalb einer Organisation nutzen, wodurch eine sehr kosteneffiziente verteilte IT-Umgebung entsteht, da in der Regel keine zusätzliche Hardware benötigt wird [Zom96]. Im Folgenden werden die wesentlichen Eigenschaften des Cluster-Computing erläutert. Weiterhin werden auf Basis formulierter Anforderungen verschiedene Cluster-Management-Systeme miteinander verglichen und ein System ausgewählt, welches bei der prototypischen Umsetzung der Parallelisierungsmethode eingesetzt wird.

4.4.1 Cluster-Computing

Der Einsatz von Supercomputern war in der Vergangenheit aufgrund der hohen Anschaffungskosten nur großen Organisationen und Forschungsinstituten vorbehalten. Durch die technologischen Weiterentwicklungen in den letzten Jahren hat sich die Leistung kommerzieller Workstations jedoch immer mehr der Leistung von Supercomputern angenähert. Dazu hat im Wesentlichen die Zunahme der Mikroprozessorleistung und der Netzwerkbandbreite beigetragen. Durch den Zusammenschluss einzelner Rechner über Hochgeschwindigkeitsnetze können somit bereits kostengünstige HPC- und HTC-Umgebungen aufgebaut und einer Vielzahl von Anwendern zugänglich gemacht werden. Der entstandene Rechnerverbund aus voneinander unabhängigen Systemen, welche über ein Netzwerk miteinander verbunden sind, wird als Cluster bezeichnet [Zom96].

Beispielhaft ist hier das System X der Virginia Polytechnic Institute and State University zu nennen. Dieser Cluster, bestehend aus 1100 Apple Power Mac Rechnern, stellte 2003 den drittschnellsten Supercomputer der Welt dar, wobei die Kosten ca. 200 Millionen Dollar unter den Kosten vergleichbarer Supercomputer lagen [Kah03]. Insbesondere bei den Kosten zeigen sich die enormen Vorteile solcher Computer-Cluster.

Computer-Cluster haben sich bereits als eine praktische und kostengünstige Ergänzung zu Supercomputern etabliert, was in erster Linie auf folgende Gründe zurückzuführen ist [Zom96]:

- Ein Cluster kann nahezu kostenneutral durch zwei oder mehr vorhandene Rechner, welche über ein Netzwerk verbunden sind, aufgebaut werden.

- Ein Cluster bietet eine leicht verfügbare Umgebung für die Forschung im parallelen Rechnen.
- Nicht verwendete Rechenkapazitäten können genutzt werden, um kostenneutral zusätzliche Rechenleistung bereitzustellen.
- Die in den Computern vorhandenen Grafikkarten können zur Bereitstellung zusätzlicher Rechenleistung genutzt werden.
- Es existiert eine Vielzahl robuster und stabiler Programme für die Verwaltung eines Clusters.

Insbesondere Workstation-Cluster bieten viele Vorteile, da ihre Leistung speziell auf die Anforderungen der jeweiligen Tätigkeiten angepasst ist. In der Produktentwicklung beinhaltet dies das interaktive Arbeiten an einem CAD- oder CAE-System, aber auch das Durchführen von numerischen Berechnungen. Dabei wird eine Workstation aber nicht ausschließlich für diese Tätigkeiten genutzt, sondern auch für andere interaktive Tätigkeiten, wie das Schreiben von E-Mails oder das Bearbeiten von Dokumenten. Während dieser interaktiven Arbeit steht eine Vielzahl ungenutzter Rechenkapazitäten zur Verfügung. Studien zeigen, dass Workstations in Unternehmen weniger als 25 % der Zeit in vollem Umfang genutzt werden und somit über 75 % an freien Rechenkapazitäten bereitstellen können. In einem Cluster können diese freien Rechenkapazitäten für die Bearbeitung rechenintensiver Probleme anderer Anwender verwendet und somit eine sehr kosteneffiziente IT-Umgebung bereitgestellt werden, was vor allem für Unternehmen eine sehr effiziente Lösung darstellt, da die Workstation bereits über das Netzwerk miteinander verbunden sind [Zom96].

Obwohl Workstation-Cluster viele Vorteile mit sich bringen, sollten sie nicht als genereller Ersatz für Supercomputer gesehen werden. Sie stellen jedoch eine gute Ergänzung zu kommerziellen HPC-Umgebungen dar. Untersuchungen in verschiedenen Rechenzentren haben gezeigt, dass viele Anwendungen, welche auf Supercomputern ausgeführt werden, dies nicht unbedingt erfordern, sondern auch auf Workstations ausgeführt werden können. Eine Kategorisierung der durchzuführenden Berechnungen sollte stets die Basis für die Bereitstellung einer IT-Umgebung bilden [Zom96].

Ein weiterer Vorteil für den Einsatz von Workstation-Clustern ist die Verfügbarkeit sehr leistungsfähiger Grafikprozessoren (Graphics Processing Unit, GPU), welche in kommerziellen Supercomputern in der Regel nicht bzw. nur bei kostenintensiver Konfiguration zur Verfügung stehen, da diese Lösungen nicht über Bildschirme für die interaktive Arbeit verfügen. So werden Grafikkartenchips für die CFD-Simulation [AD11], [KTAG10] sowie für die FEM-Simulation [ACD⁺12], [MS11] verwendet. Auch für die Topologieoptimierung können Grafikkartenchips verwendet werden [CRG14], [ZP13]. Neben der Durchführung von Sensitivitätsanalysen mit Hilfe von Monte-Carlo-Simulation [JGG⁺11] können auch ganze Parameteroptimierungen auf Grafikkartenchips durchgeführt werden. Hierbei werden bereits verschiedene Optimierungsverfahren wie Evolutionäre und Genetische Algorithmen [SD13], [HLF13], das Simulierte Ausglühen (Simulated Annealing) [Zbi12], [FGLSV13], Ameisenkolonialgorithmen [UIN13], Partikelschwarmalgorithmen [WF12] oder die Tabu-Suche [CB11] angewendet. Aufgrund der großen Anzahl von Grafikchips kann bei einfachen Evaluationen eine zusätzliche Parallelisierung auf der Grafikkarte erfolgen. Auch für die Generierung von Metamodellen lassen sich die Chips einer Grafikkarte nutzen [Toa15].

Ein Workstation-Cluster stellt also eine sehr kostengünstige, umfangreiche und individuell anpassbare IT-Umgebung für das Bearbeiten von Optimierungsaufgaben in der Produktentwicklung bereit.

4.4.2 Auswahl eines Cluster-Management-Systems

Zur Realisierung des in Abschnitt 4.4.1 beschriebenen Workstation-Clusters zur parallelen Bearbeitung von Optimierungsaufgaben wird ein Cluster-Management-System benötigt, welches die Verwaltung des Clusters und dessen Ressourcen sowie die Zuweisung von Berechnungsjobs übernimmt. Da die Zuweisung der Berechnungsjobs das Erstellen eines Ablaufplans (engl. schedule) beinhaltet, werden Cluster-Management-Systeme auch als Scheduling-Systeme oder Scheduler bezeichnet. Weiterhin werden auch oft die Begriffe Lastverteilungssystem und Workload-Management-System verwendet, da mittels dynamischer Allokation (s. Abschnitt 4.1) die Belastung durch die Berechnungsjobs auf die Rechner des Clusters verteilt wird.

Zur Auswahl eines geeigneten Systems wird zunächst eine Anforderungsliste erstellt. Diese dient insbesondere dazu, die Forderungen und die Wünsche an das System zu konkretisieren und zu kategorisieren [FG13]. Die Anforderungen werden in folgende Kategorien eingeteilt:

- Plattform
- Lizenzierung
- Benutzungsfreundlichkeit
- Rechnermanagement
- Jobmanagement
- Ressourcenmanagement
- Energiemanagement
- Lizenzmanagement
- Sicherheit
- Monitoring
- Zusätzliche Funktionen

Die wichtigste Anforderung an das Cluster-Management-System ist die Plattform. Da in dem für die prototypische Umsetzung genutzten Rechner-Pool ausschließlich das Betriebssystem Windows vorhanden ist, muss die Kompatibilität zwischen den Systemen gewährleistet sein. Weiterhin soll das Cluster-Management-System kostenneutral implementiert werden, wobei ein offener Quellcode bei der Lizenzierung bevorzugt wird, um eventuelle spätere Anpassungen oder Erweiterungen der Software selbstständig vornehmen zu können. Bei der Benutzungsfreundlichkeit wird zwischen der Installation und Konfiguration, der Nutzung sowie der Dokumentation unterschieden. Eine grafische Benutzungsoberfläche stellt hierbei einen Wunsch an das System dar. Die Steuerung über

die Kommandokonsole muss mindestens möglich sein. Das System soll zudem eine heterogene IT-Umgebung unterstützen. Dabei sollen die Rechner gruppiert und priorisiert werden können.

Da der zu realisierende Workstation-Cluster weiterhin die interaktive Arbeit auf den Rechnern ermöglichen und diese auch in keinsten Weise einschränken soll, was als einer der wesentlichen Vorteile eines Workstation-Clusters gilt [Zom96], soll das Cluster-Management-System ein flexibles Job- und Ressourcenmanagement ermöglichen. Dies beinhaltet das Pausieren und Fortsetzen von laufenden Jobs und die Verfügbarkeit der Rechner gemäß ihrer Nutzung, der installierten Software und der zur Verfügung stehenden Lizenzen. Auch das Energiemanagement, die Erfüllung von Sicherheitsanforderungen und die Überwachung des Clusters spielen bei der Systemauswahl eine wesentliche Rolle. Zusätzliche Funktionen wie diverse Anwendungsprogrammierschnittstellen sowie die Nutzung von Virtualisierungs- und Cloud-Techniken bilden untergeordnete Anforderungen. Die vollständige Anforderungsliste ist in Tabelle A.4 des Anhangs A.4 dargestellt.

Die auf Basis der Anforderungsliste recherchierten Cluster-Management-Systeme sind in Tabelle A.5 im Anhang A.4 mit ihren jeweiligen Eigenschaften sowie Vor- und Nachteilen aufgelistet. Bereits im Vorfeld können die Systeme *DTCCondor* und *schedulix* für die Realisierung ausgeschlossen werden. DTCCondor ist speziell für die Verwendung an der Universität Liverpool entwickelt und stellt gemäß der formulierten Anforderungen keinen Mehrwert gegenüber dem System HTCondor dar. Das System *schedulix* wird aufgrund der geringen Dokumentation sowie des hohen administrativen Aufwandes ebenfalls im Vorfeld ausgeschlossen.

Die eingegrenzten recherchierten Cluster-Management-Systeme werden in einem binären Vergleich hinsichtlich der Kriterien Funktionalität, Benutzbarkeit und Umsetzbarkeit miteinander verglichen. Dabei wird das als besser eingestufte System mit 1 bewertet, das als schlechter bewertete System mit 0. Die detaillierten Ergebnisse des binären Vergleichs sind im Anhang A.4 zu finden (Tabellen A.6 – A.8). Die aufsummierten Teilergebnisse des binären Vergleichs sind in Abbildung 4.9 dargestellt.

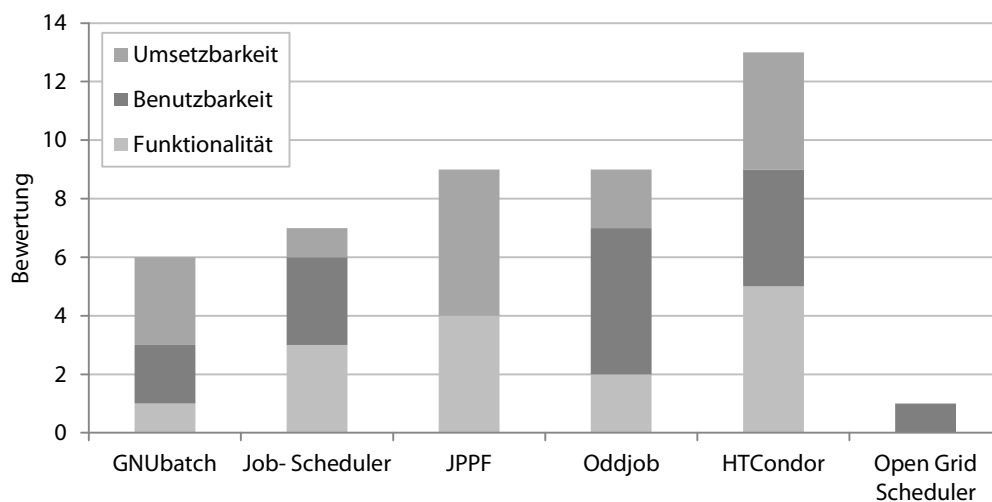


Abbildung 4.9: Ergebnis des binären Vergleiches windows-basierter Cluster-Management-Systeme, nach [Dzi15]

Das Cluster-Management-System HTCondor erzielt nicht in jedem der drei Kriterien die beste Bewertung, wird jedoch in der Funktionalität am besten bewertet und stellt hin-

sichtlich der drei Kriterien die beste Gesamtlösung dar. Vor allem aufgrund des hohen Funktionsumfangs und der Anpassbarkeit ist es sehr gut für die Realisierung eines Clusters zur prototypischen Umsetzung geeignet. Das System unterstützt ein flexibles Ressourcen- und Jobmanagement in heterogenen Umgebungen, ist Open-Source lizenziert, verfügt über eine umfangreiche Dokumentation und kann über einfache Batch-Programmierung und Skripte gesteuert werden [Cen15]. Als nachteilig werden die fehlende Benutzeroberfläche und die Überwachung des Clusters durch externe Programme bewertet [Dzi15].

4.5 Prioritätenbasierte Bearbeitung

Im Folgenden wird die prioritätenbasierte Bearbeitung von Berechnungsaufgaben (Jobs) in einem Cluster-Management-System beschrieben, welches die Zuweisung der Jobs übernimmt. Bereits anhand von einfachen Beispielen kann aufgezeigt werden, welche Effizienzsteigerung durch die prioritätenbasierte Bearbeitung erzielt werden kann. Dazu werden zunächst drei Berechnungsjobs (Job A, Job B, Job C) mit unterschiedlichen Laufzeiten t definiert, welche als Teil eines parallel verarbeiteten Evaluationsprozesses einer Optimierung angesehen werden.

Die Jobs werden über eine For-Schleife an das Scheduling-System übermittelt. Hier wird über eine Liste („individuals“), welche drei Elemente enthält, iteriert. Jedes Element („individual“) stellt eine Produktvariante innerhalb der Optimierung dar. Der Quellcode des Übergabeprozesses ist folgend vereinfacht dargestellt.

```

1 for individual in individuals:
2     condor_submit Job A
3     condor_submit Job B
4     condor_submit Job C

```

Durch die Übermittlung der Jobs an das Cluster-Management-System werden diese der Warteschlange (Queue) hinzugefügt und es wird vom System ein Ablaufplan erstellt. Dieser Prozess wird als Scheduling bezeichnet [EBS15]. Dabei werden gemäß der zu evaluierenden Produktvarianten der Optimierung verschiedene Instanzen der jeweiligen Jobs erzeugt. Abbildung 4.10 stellt diesen Prozess schematisch dar.

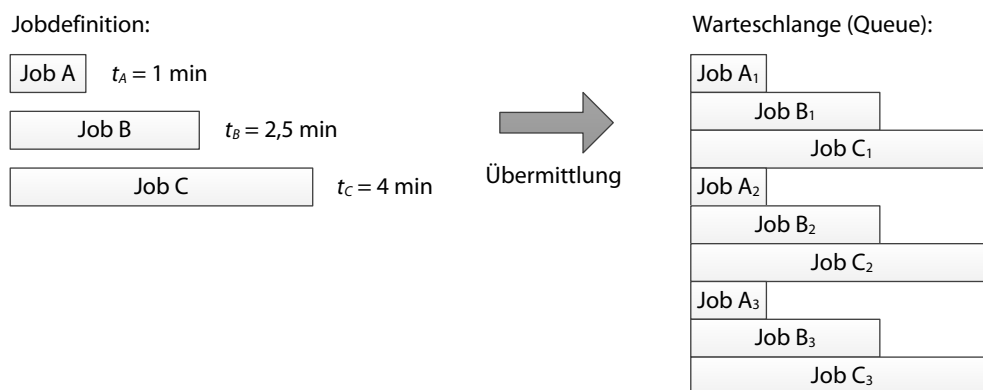


Abbildung 4.10: Übermittlung von Jobs an die Warteschlange

Die Warteschlange repräsentiert eine abstrakte Datenstruktur und beinhaltet alle durch eine Anwendung oder das Betriebssystem initialisierten Prozesse bzw. Jobs in serialisierter Form, welche bis zu ihrer Bearbeitung in einen Wartezustand versetzt werden [FH11].

Sofern nicht anders festgelegt, werden die Jobs in der Warteschlange nach dem Prinzip des First In - First Out (FIFO) bearbeitet. Da hierbei lediglich die stapelartige Anordnung der Prozesse in der Warteschlange berücksichtigt wird, stellt diese Form eine robuste und bewährte Vorgehensweise der Bearbeitung von Berechnungsjobs dar [RB13]. Insbesondere bei vielen voneinander abhängigen Prozessen unterschiedlicher Länge, wie es bei der Parallelisierung von Evaluationsprozessen einer Optimierung der Fall ist, liefert diese Vorgehensweise jedoch nicht die kürzeste Laufzeit. Wie in Abbildung 4.11 zu sehen ist, kann die Laufzeit durch eine prioritätenbasierte Anordnung der Prozesse verkürzt werden.

First In - First Out:

Slot 1:

Job A ₁	Job C ₁	Job C ₂	Job C ₃
--------------------	--------------------	--------------------	--------------------

 $t = 13 \text{ min}$

Slot 2:

Job B ₁	Job A ₂	Job B ₂	Job A ₃	Job B ₃
--------------------	--------------------	--------------------	--------------------	--------------------

$\Delta t = 1,5 \text{ min} \approx 11,5 \%$

Prioritätenbasiert:

Slot 1:

Job C ₁	Job C ₃	Job B ₃	Job A ₃	1,5 min
--------------------	--------------------	--------------------	--------------------	---------

 $t = 11,5 \text{ min}$

Slot 2:

Job C ₂	Job B ₁	Job B ₂	Job A ₁	Job A ₂
--------------------	--------------------	--------------------	--------------------	--------------------

(a) Keine Abhängigkeit und keine Restriktion

First In - First Out:

Slot 1:

Job A ₁	Job B ₁	Job B ₂	Job A ₃	Job B ₃
--------------------	--------------------	--------------------	--------------------	--------------------

Slot 2:

Job A ₂	Job C ₁	Job C ₂	Job C ₃
--------------------	--------------------	--------------------	--------------------

 $t = 13 \text{ min}$

Prioritätenbasiert:

$\Delta t = 1,5 \text{ min} \approx 11,5 \%$

Slot 1:

Job A ₁	Job A ₃	Job C ₂	Job B ₁	Job B ₂
--------------------	--------------------	--------------------	--------------------	--------------------

Slot 2:

Job A ₂	Job C ₁	Job C ₃	Job B	1,5 min
--------------------	--------------------	--------------------	-------	---------

 $t = 11,5 \text{ min}$

(b) Abhängigkeit A - BC und keine Restriktion

First In - First Out:

Slot 1:

Job A ₁	Job B ₁	Job A ₂	Job B ₂	Job A ₃	Job B ₃
--------------------	--------------------	--------------------	--------------------	--------------------	--------------------

Slot 2:

Job C ₁	Job C ₂	Job C ₃
--------------------	--------------------	--------------------

 $t = 13 \text{ min}$

Prioritätenbasiert:

$\Delta t = 1 \text{ min} \approx 7,7 \%$

Slot 1:

Job A ₁	Job A ₂	Job A ₃	Job C ₂	Job B ₁	Job B ₃	1 min
--------------------	--------------------	--------------------	--------------------	--------------------	--------------------	-------

 $t = 12 \text{ min}$

Slot 2:

Job C ₁	Job C ₃	Job B ₂
--------------------	--------------------	--------------------

(c) Abhängigkeit A - BC und Restriktion 1 x A

Abbildung 4.11: Reduzierung der Laufzeit durch prioritätenbasierte Bearbeitung bei der Verwendung von zwei Berechnungsslots

Existieren keine Abhängigkeiten und keine externen Restriktionen der Jobs (a) kann in dem dargestellten Beispiel die Laufzeit t durch prioritätenbasierte Bearbeitung von 13 min auf 11,5 min reduziert werden. Die Prioritäten der zu bearbeitenden Jobs werden auf Basis ihrer jeweiligen Laufzeit festgelegt. Längere Jobs erhalten eine höhere Priorität als kürzere Jobs und werden somit zuerst ausgeführt. Somit kann die Laufzeit um 1,5 min reduziert werden (s. grauer Block in Abbildung 4.11). Dies entspricht einer Laufzeitreduzierung von 11,5 %.

Das gleiche Ergebnis wird unter Berücksichtigung einer Abhängigkeit der Jobs (b) erzielt. Hier hängen die Jobs B und C vom Job A ab und können nicht vor dessen Fertigstellung beginnen. Durch die prioritätenbasierte Bearbeitung wird ebenfalls eine relative Laufzeitverkürzung von 11,5 % erreicht.

Wird eine zusätzliche Restriktion von Job A berücksichtigt, nach der Job A nur einmal zu einem Zeitpunkt ausgeführt werden darf und somit nicht parallel bearbeitet wird, kann die Laufzeit um 1 min verkürzt werden, was einer Reduzierung um 8,3 % entspricht. Eine solche Restriktion kann in der Realität durch das Vorhandensein einer bestimmten Anzahl von Softwarelizenzen hervorgerufen werden.

Anhand dieses Beispiels wird der Nutzen der prioritätenbasierten Bearbeitung ersichtlich. Eine solche Vorgehensweise ist in jedem Fall anzustreben. Die Verwendung von drei Slots zur Bearbeitung der Jobs führt zu ähnlichen Ergebnissen. Dabei wird in den Fällen (a) und (b) eine Laufzeitreduzierung von 16,7 % erzielt. Lediglich unter der Berücksichtigung der Restriktion von Job A bleibt die Laufzeit konstant (c). Die Laufzeitreduzierung unter Verwendung von drei Slots ist in Abbildung A.2 im Anhang A.3.1 dargestellt.

Die Ermittlung der optimalen Prozessprioritäten zur Minimierung der Gesamtlaufzeit kann den Scheduling-Problemen zugeordnet werden, wobei in klassischen Scheduling-Problemen von statischer Lastverteilung bzw. Allokation ausgegangen wird [BBKS15] (s. Abschnitt 4.1). Dabei steht die optimale Zuordnung von Jobs und Slots mit dem Ziel im Vordergrund, stets die geringste Gesamtlaufzeit zu erreichen. Scheduling-Probleme zählen zur Komplexitäts- bzw. Problemklasse der NP-Probleme (nichtdeterministisch polynomiale Zeit). Probleme dieser Klasse wachsen mit der Problemgröße in ihrer Komplexität überexponentiell an und lassen sich mit einem deterministischen Ansatz nicht in Polynomialzeit lösen. Der Begriff der Polynomialzeit wird hierbei verwendet, da unter der Anwendung eines deterministischen Lösungsansatzes die zur Lösung benötigte Rechenzeit mit der Problemgröße stärker ansteigt, als durch eine Polynomfunktion beschreibbar ist. Da somit deterministische Ansätze zur Problemlösung nicht effizient sind, werden NP-Probleme näherungsweise durch nichtdeterministische Ansätze und Heuristiken gelöst [FH11].

Für das Scheduling von Jobs in einer verteilten Umgebung existiert eine Vielzahl unterschiedlicher Verfahren, welche gemäß ihrer Funktionsweise in drei Klassen eingeteilt werden. Diese Klassen sind mit den jeweiligen Subklassen in Abbildung 4.12 dargestellt und werden im Folgenden erläutert.

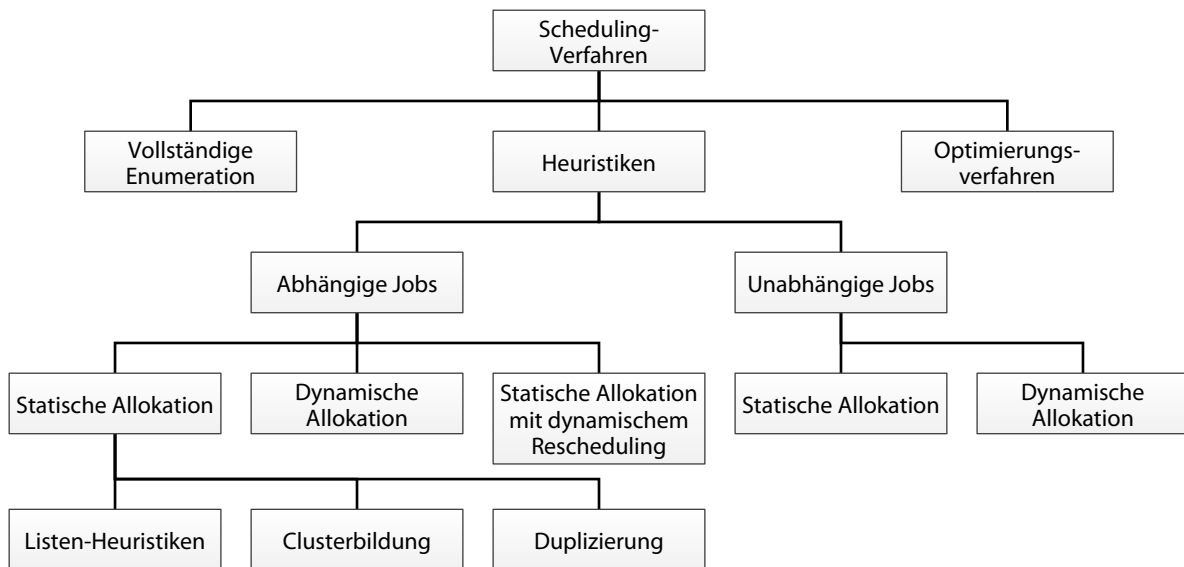


Abbildung 4.12: Klassifizierung von Scheduling-Verfahren, nach [THM02], [DA06]

Vollständige Enumeration: Bei der vollständigen Enumeration werden alle möglichen Kombinationen des Problems überprüft und durch einen Vergleich der Ergebnisse die optimale Lösung gefunden. Das Verfahren ist somit mit einem vollfaktoriellen Versuchsplan gleichzusetzen (s. Abschnitt 2.7.1). Aufgrund des sehr hohen numerischen Aufwands eignet sich die vollständige Enumeration im praktischen Einsatz nur für kleine Probleme [Gab16]. Im weiteren Verlauf dieser Arbeit wird dieses Verfahren verwendet, um einmalig die optimalen Prozessprioritäten zu ermitteln und somit die Eignung etablierter Heuristiken und Optimierungsverfahren zur Ermittlung der optimalen Prozessprioritäten zu überprüfen (s. Abschnitt 5.4.2).

Heuristiken: Heuristiken (griech. Anweisung) beschreiben ermittelte zuverlässige Lösungen basierend auf der zugrunde liegenden komplexen Struktur eines Problems. Im Gegensatz zu Optimierungsverfahren wird die Näherungslösung nicht iterativ, sondern direkt ermittelt [FH11]. Heuristiken für das Scheduling werden nach dem Vorhandensein von Abhängigkeiten zwischen den Jobs sowie nach statischer und dynamischer Allokation unterschieden [DA06]. Da die dynamische Allokation der Jobs in der verteilten Umgebung durch HTCCondor erfolgt und die statische Zuweisung der optimalen Jobprioritäten im Vordergrund steht, wird sich auf Heuristiken zur Ermittlung der optimalen Jobprioritäten beschränkt. Diese Heuristiken werden auch als Listen-Heuristiken bezeichnet. Dabei dient analog zum eingangs beschriebenen Beispiel eine Liste von Jobs, die Warteschlange bzw. Queue, als Ausgangsbasis für das Scheduling [DA06].

Viele Listen-Heuristiken haben ihren Ursprung im Projektmanagement und dienen als Vorlage für die Priorisierung von Berechnungsjobs. Folgend werden die wichtigsten Listen-Heuristiken beschrieben:

- First In - First Out (FIFO): Die Jobs werden gemäß des Zeitpunktes der Aufnahme in die Warteschlange verarbeitet [TA10]. Es wird lediglich die stapelartige Anordnung der Jobs in der Warteschlange berücksichtigt. Das FIFO-Prinzip dient in weiteren Betrachtungen der Ermittlung des Referenzwertes zur Bestimmung der erzielten Laufzeitverkürzung.

- Shortest Processing Time (SPT): Jobs mit einer kürzeren Laufzeit erhalten eine höhere Priorität und werden längeren Jobs vorgezogen [TA10]. Da sich somit am Ende der Warteschlange die längeren Jobs sammeln, kann Leerlauf in einigen Slots entstehen, wenn die letzten längeren Jobs bearbeitet werden.
- Longest Processing Time (LPT): Die Zuweisung der Jobprioritäten erfolgt gemäß der jeweiligen Laufzeit. Je länger die Laufzeit eines Jobs ist, desto höher wird dieser priorisiert. Somit werden längere Jobs kürzeren vorgezogen und die kürzeren Jobs am Ende der Warteschlange gesammelt. Diese können dann flexibel auf die Slots verteilt werden, wodurch die gesamte Bearbeitung am Ende homogenisiert und möglicher Leerlauf reduziert wird.
- Most Work Content Remaining (MWR): Die Laufzeit eines Jobs wird mit den Laufzeiten der von diesem Job direkt und indirekt abhängigen Jobs addiert. Je höher die summierte Laufzeit, desto höher wird die Priorität festgelegt. Somit werden Jobs, von denen andere Jobs mit langer Laufzeit abhängen, bevorzugt [Fün06].
- Longest Path Following (LPF): Dieses Verfahren ist analog zum Verfahren des *kritischen Pfades* [Hu61] zu sehen. Hierbei werden die Laufzeiten entlang eines Pfades abhängiger Jobs aufsummiert und der Pfad mit der höchsten summierten Laufzeit am höchsten priorisiert [Fün06].
- Most Total Successors (MTS): Die Priorisierung erfolgt gemäß der Anzahl der direkt und indirekt abhängigen Jobs. Je höher die Anzahl der abhängigen Jobs, desto höher wird der Job priorisiert [Fün06].
- Heterogeneous Earliest Finish Time (HEFT): Bei diesem speziell für das Scheduling von Jobs auf CPU entwickelten Verfahren wird die Laufzeit eines Jobs mit den Laufzeiten der nachfolgenden abhängigen Jobs addiert, wobei zusätzlich die jeweiligen Overhead-Zeiten berücksichtigt werden. Je höher die aufsummierte Zeit, desto höher wird der Job priorisiert. Der HEFT-Algorithmus beinhaltet zudem die Zuweisung der Jobs durch statische Allokation. Dabei wird stets der Slot ausgewählt, welcher den frühesten Zeitpunkt der Fertigstellung eines Jobs gewährleistet. Zusätzlich können Jobs auch in Freiräumen zwischen zwei bereits zugewiesenen Jobs eingeordnet werden [THM02].

Aufgrund seiner Spezialisierung ist der HEFT-Algorithmus sehr effizient und liefert eine gute Ergebnisqualität [WPF05]. Es werden jedoch ausschließlich CPU als Ressourcen betrachtet und die Allokation erfolgt statisch, was bei einem Ressourcenausfall nach der Zuweisung zu Engpässen führen kann.

Optimierungsverfahren: Die Lösung von Scheduling-Problemen bzw. die Ermittlung von Prioritäten durch Optimierungsverfahren stellt im Gegensatz zu Heuristiken eine iterative Lösung dar. Dabei kommen aufgrund der diskreten, kombinatorischen Problemstruktur vorwiegend stochastische Optimierungsverfahren zum Einsatz, z. B. Genetische Algorithmen, Particle Swarm oder Simulated Annealing (s. Abschnitt 2.6.2). Durch Optimierungsverfahren lässt sich die gleiche Ergebnisqualität wie durch den HEFT-Algorithmus erzielen [THM02], [WPF05], [RGGB07], [KV16]. Da diese Optimierungsverfahren keine reinen Speziallösungen darstellen, sind sie flexibel an die jeweilige Problemstellung anpassbar. Nachteilig ist die hohe Anzahl an Evaluationen, welche bei sehr kurzen Evaluationszeiten jedoch vernachlässigt werden kann.

4.6 Abschließende Bemerkungen

In diesem Kapitel wurde die effiziente Bearbeitung von Optimierungsaufgaben in der Produktentwicklung durch die Parallelisierung von Evaluationsprozessen betrachtet. Bei der Analyse konventioneller Parallelisierungsmethoden wurde festgestellt, dass diese nur selten die verwendeten Ressourcen sowie die Limitierung dieser Ressourcen berücksichtigen. Die vorhandene Software und die zur Verfügung stehenden Softwarelizenzen finden keine Berücksichtigung.

Da zur Evaluation der Produktvarianten bei einer Optimierung in der Regel kommerzielle Software verwendet wird, deren Verfügbarkeit den Grad der Parallelisierung und somit die Effizienz der Optimierung einschränken kann, wurde die konventionelle Ressourcendefinition erweitert und unter den Begriffen Concurrent und Simultaneous Optimization eine neue Methode zur effizienten und dynamischen Parallelisierung von Evaluationsprozessen von Optimierungen in der Produktentwicklung vorgestellt.

Diese Methode wird im folgenden Kapitel bei der Entwicklung eines Frameworks zur effizienten und dynamischen Parallelisierung prototypisch umgesetzt. Die verteilte IT-Umgebung wird dabei durch einen Workstation-Cluster unter Verwendung des Cluster-Management-Systems HTCCondor realisiert.

Weiterhin wurde in diesem Kapitel herausgestellt, wie durch die prioritätenbasierte Bearbeitung von Prozesselementen bei gleichem Ressourceneinsatz eine Verkürzung der Laufzeit und somit eine Steigerung der Effizienz erzielt werden kann. Zur Ermittlung der optimalen Prioritätswerte, welche bei der Bearbeitung von Prozesselementen in der kürzesten Laufzeit resultieren, wurden verschiedene Scheduling-Verfahren vorgestellt.

Bei der Analyse der Scheduling-Verfahren zeigt sich, dass in erster Linie jeweils nur eine Art von Ressourcen berücksichtigt wird, z. B. CPU eines oder verschiedener Rechner (s. Tabelle A.3 im Anhang A.3.2). Diese eindimensionale Ressourcendefinition ist für die effiziente und dynamische Parallelisierung von Optimierungsprozessen nicht ausreichend, da neben der Hardware auch die verfügbare Software und Lizenzen berücksichtigt werden müssen. Daher kann der etablierte HEFT-Algorithmus für die weitere Verwendung in dieser Arbeit ausgeschlossen werden.

Die Verwendung weiterer Scheduling-Verfahren wird im Rahmen der prototypischen Umsetzung im folgenden Kapitel anhand von verschiedenen Testszenarien untersucht. Neben der vollständigen Enumeration, werden dabei die Heuristiken First In - First Out (FIFO), Longest Path Following (LPF) und Longest Processing Time (LPT) sowie verschiedene Optimierungsverfahren betrachtet.

5 Prototypische Umsetzung

Innerhalb der prototypischen Umsetzung wird ein Framework entwickelt, mit dem die in Kapitel 4 beschriebene Methode zur effizienten und dynamischen Parallelisierung von Evaluationsprozessen von Optimierungen in der Produktentwicklung realisiert wird. Das zu entwickelnde Framework soll dabei verschiedene Funktionen und Verfahren zur effizienten Durchführung von Optimierungen und DoE-Studien in einem Cluster beinhalten.

Zunächst werden in diesem Kapitel Anforderungen an das Framework formuliert. Da dieses primär aus einem Optimierungssystem und einem Cluster besteht, werden die Anforderungen ebenfalls in diesen Kategorien unterschieden. In einer dritten Kategorie werden Anforderungen an die Benutzungsfreundlichkeit des Frameworks formuliert, um somit dessen spätere Nutzung zu erleichtern. Ausgehend von den Anforderungen erfolgt die Konzeptionierung des Frameworks. Dabei wird ein Aktivitätsdiagramm der Durchführung einer Optimierung erstellt, um somit die wesentlichen Vorgänge bei der Durchführung einer dynamisch parallelen Optimierung zu identifizieren und diese bei der weiteren Entwicklung des Frameworks zu berücksichtigen. Weiterhin werden die in das Framework integrierten Systeme, die jeweiligen Aufgaben dieser Systeme und die Interaktionen zwischen den Systemen spezifiziert. Dies bildet das Konzept des Frameworks.

Auf Basis des Konzeptes erfolgt die Realisierung der Anforderungen. Zunächst werden die Integration und Konfiguration des Clusters erläutert. Anschließend erfolgt die Beschreibung des entwickelten Optimierungssystems DAG2OPT. Dieses System dient zur Definition des Evaluationsmodells einer Optimierung durch einen gerichteten azyklischen Graph (Directed Acyclic Graph, DAG) und zur Durchführung der Optimierung in dem Cluster.

Zudem werden in DAG2OPT die Prozessprioritäten zur Realisierung der prioritätenbasierten Bearbeitung der Prozesselemente festgelegt. Die in Abschnitt 4.5 beschriebenen Verfahren zur Bestimmung der optimalen Prozessprioritäten werden hierbei in drei Test-szenarien bezüglich ihrer Effektivität miteinander verglichen. Anhand der Effektivität wird eine Methode ausgewählt und in DAG2OPT implementiert. Dies ermöglicht die Ermittlung der optimalen Prioritätswerte der Prozesselemente des Evaluationsprozesses im Vorfeld einer Produktoptimierung.

Im Anschluss an die Entwicklung erfolgt die Validierung des Frameworks anhand von drei Fallstudien, welche charakteristische multidisziplinäre Optimierungsaufgaben aus realen Entwicklungsprojekten darstellen. Aus den Erkenntnissen dieser Fallstudien werden abschließend Handlungsempfehlungen zur effizienten Durchführung einer Optimierung abgeleitet.

5.1 Anforderungen

Die Entwicklung des Frameworks erfolgt auf der Basis von Anforderungen, welche in diesem Abschnitt definiert und beschrieben werden. Da die Umsetzung der hier definierten Anforderungen selbstständig erfolgt und keinen Restriktionen durch externe Einflüsse unterliegt, wird im Gegensatz zur reinen Auswahl des Cluster-Management-Systems (s. Abschnitt 4.4.2) nicht explizit zwischen Forderungen und Wünschen unterschieden. Bei der

Auswahl und Formulierung der Anforderungen wird sich an den Anforderungen an Optimierungssysteme von Schumacher [Sch13] orientiert, da diese eine fundierte Basis zur Beurteilung der Funktionalität von Optimierungssystemen darstellen.

Das zu entwickelnde Framework besteht im Wesentlichen aus einem Optimierungssystem und einem Cluster. Eine detaillierte Beschreibung der in das Framework integrierten Systeme erfolgt in Abschnitt 5.2. Weiterhin wird bei der Entwicklung des Frameworks die Benutzungsfreundlichkeit betrachtet. Die Anforderungen an das Framework werden somit in drei Kategorien eingeordnet: Anforderungen an die wesentlichen Funktionen des Optimierungssystems, Anforderungen an die Benutzungsfreundlichkeit und Anforderungen an die Integration und Verwendung des Clusters.

Bezüglich der wesentlichen Funktionen des Optimierungssystems sind folgende Anforderungen zu berücksichtigen:

- **Flexible Modellierung des Evaluationsmodells:** Die zur Evaluation benötigten Prozesse sollen modular und flexibel in das Evaluationsmodell integrierbar sein. Dabei muss es möglich sein, zwischen parallelen und sequentiellen Prozesselementen zu unterscheiden.
- **Modellkomposition und parallele Verarbeitung:** Nach dem Ansatz des Concurrent und Simultaneous Optimization sollen flexibel alle Arten der Modellkomposition und Parallelisierung realisiert werden.
- **Editoren zur Anpassung der verwendeten Skripte:** Die zur Evaluation verwendeten Skripte sollen direkt in dem Optimierungssystem über Editoren bearbeitet werden können.
- **Offene Schnittstellen zu CAx-Systemen:** Die zur Evaluation verwendeten CAx-Systeme sollen in das Optimierungssystem integrierbar sein. Dazu vorgesehene Schnittstellen sollen flexibel anpassbar und erweiterbar sein.
- **Batch-Fähigkeit:** Damit das erstellte Optimierungs- und Evaluationsmodell in übergeordnete Prozesse integriert werden kann, soll die Optimierung auch automatisch über Batch-Skripte gestartet werden können.
- **Integration vorhandener Optimierungs- und Evaluationsmodelle:** Externe Optimierungs- und Evaluationsmodelle sollen als untergeordnete Prozesse in die Optimierung integriert werden können.
- **Offenheit des Systems:** Das zu entwickelnde System soll über offene Schnittstellen flexibel anpassbar und erweiterbar sein.
- **Verschiedene Optimierungsalgorithmen:** Das System soll die Anwendung unterschiedlicher Optimierungsalgorithmen unterstützen. Zudem sollen auch eigene Optimierungsalgorithmen implementiert werden können.
- **Explorative Untersuchung des Lösungsraumes und Metamodelle:** Die explorative Untersuchung des Lösungsraumes durch statistische Versuchspläne soll durch das System unterstützt werden. Somit kann der untersuchte Lösungsraum durch Metamodelle approximiert werden. Dabei ist auch die Verwendung extern erstellter Versuchspläne und Metamodelle zu berücksichtigen.

- **Restart-Funktion:** Laufende Optimierungen sollen angehalten und neu gestartet werden können. Der Neustart sollte in jedem Fall möglich sein, sobald Zwischenergebnisse vorhanden sind.
- **Diskrete Werte der Designvariablen:** Die Verwendung diskreter Werte der Designvariablen soll unterstützt werden, da z. B. Halbzeuge und Werkzeuge oft nur in definierten Abmessungen verfügbar sind und somit die Realisierung des optimierten Produktes erleichtert wird.
- **Verwendung mathematischer Funktionen:** Mathematische Funktionen sollen frei erstellt und editiert werden können, z. B. zur Modifikation und Berücksichtigung der berechneten Zustandsvariablen in der Zielfunktion. Dabei soll auch die Verwendung von Filtern und Vergleichsmöglichkeiten unterstützt werden.
- **Benutzerdefinierte Archivierung der Ergebnisdateien:** Die Archivierung der bei der Optimierung erzeugten Ergebnisdateien soll zur Kontrolle des verwendeten Speicherplatzes durch den Anwender editierbar sein.

Neben den wesentlichen Funktionen des Optimierungssystems soll bei der Entwicklung auch die Benutzungsfreundlichkeit des Systems berücksichtigt werden. Dadurch sollen notwendige Programmierkenntnisse reduziert und die Arbeit mit dem System erleichtert werden. Da es sich bei der Entwicklung des Frameworks um eine prototypische Realisierung handelt, werden Anforderungen des Supports und der Dokumentation vernachlässigt. Hinsichtlich der Benutzungsfreundlichkeit werden folgende Anforderungen betrachtet:

- **Grafische Benutzungsoberfläche:** Die Kernfunktionen des zu entwickelnden Systems sollen über eine grafische Benutzungsoberfläche (Graphical User Interface, GUI) gesteuert werden. Lediglich weiterführende Funktionen, wie die Anpassung und Erweiterung des Systems, sollen über Veränderungen des Quellcodes erfolgen.
- **Überwachung des Optimierungsverlaufs:** Während einer laufenden Optimierung sollen der Verlauf und der Fortschritt der Optimierung kontrolliert und die Optimierung bei auftretenden Fehlern abgebrochen und angepasst werden können.
- **Ausgabe und grafische Aufbereitung der Ergebnisse:** Der Verlauf und die Ergebnisse der Optimierung sollen für die externe Weiterverarbeitung im ASCII-Format ausgelesen werden können. Dabei soll die Darstellung der Ergebnisse in zwei- und dreidimensionalen Diagrammen möglich sein. Zudem soll die Darstellung der Designvariablenwerte in Abhängigkeit der Optimierungsschritte unterstützt werden.

Die Integration des Clusters erfolgt mittels des Cluster-Management-Systems HTCondor, welches hinsichtlich der Auswahlkriterien Umsetzbarkeit, Benutzbarkeit und Funktionalität die beste Gesamtlösung der untersuchten Systeme darstellt (s. Abschnitt 4.4.2). Da die Integration einen wesentlichen Bestandteil der prototypischen Umsetzung darstellt, werden die Anforderungen der in Abschnitt 4.4.2 erstellten Anforderungsliste, welche nicht direkt durch HTCondor abgebildet werden oder speziell zur Bearbeitung von Optimierungsaufgaben angepasst werden müssen, extrahiert und gesondert umgesetzt. Für die Integration und Verwendung des Clusters ergeben sich somit folgende Anforderungen:

- **Definition und Übermittlung einzelner Jobs:** Die Definition und Übermittlung einzelner Berechnungsjobs erfolgt innerhalb des zu entwickelnden Frameworks. Neben der Durchführung von Optimierungen sollen zudem auch andere Berechnungsjobs an den Cluster übermittelt werden können.

- **Definition und Übermittlung zusammenhängender Jobs:** Bei der Definition von Berechnungsjobs sollen auch zusammenhängende Jobs berücksichtigt werden, welche Abhängigkeiten untereinander aufweisen.
- **Monitoring von Ressourcen und Jobs:** Das Monitoring und Editieren der Ressourcen des Clusters sowie der in dem Cluster bearbeiteten Jobs soll gewährleistet sein.
- **Priorisierung von Jobs:** Das System soll die prioritätenbasierte Bearbeitung von Jobs unterstützen (s. Abschnitt 4.5).
- **Fehlerbehandlung von Lizenzfehlern:** Treten Lizenzfehler auf, z. B. durch einen Ausfall des Lizenzservers, sollen laufende Berechnungsjobs nicht abgebrochen und gelöscht werden, sondern es soll eine Fehlerbehandlung durchgeführt werden, durch welche diese Berechnungsjobs ggf. nach einer definierten Wartezeit erneut gestartet werden.
- **Flexible Verwaltung von Ressourcen:** Die zur Verfügung stehenden Ressourcen sollen durch das System flexibel verwaltet werden und je nach den Anforderungen der Berechnungsjobs bereitgestellt werden. Dabei soll interaktives Arbeiten an den Rechnern des Clusters jederzeit möglich sein und die interaktive Arbeit nicht beeinträchtigt werden.
- **Priorisierung von Ressourcen:** Zur Verfügung stehende Ressourcen sollen durch den Anwender priorisiert werden können und höher priorisierte Ressourcen bei der Bearbeitung von Jobs bevorzugt werden.
- **Energieverwaltung:** Nicht verwendete Rechner des Clusters sollen automatisch einen Zustand niedrigeren Energieverbrauchs einnehmen und bei Bedarf wieder in den normalen Betriebszustand wechseln.

Diese Anforderungen stellen die Grundlage für die prototypische Entwicklung des Frameworks dar. Deren Umsetzung erfolgt in den nachfolgenden Abschnitten.

5.2 Konzeptionierung

Ausgehend von den in Abschnitt 5.1 beschriebenen Anforderungen erfolgt die Konzeptionierung des zu entwickelnden Frameworks. Dazu wird zunächst ein Aktivitätsdiagramm des Optimierungssystems zur Durchführung einer Optimierung erstellt. Die Abbildung der Aktivitäten ist eine Modellierungstechnik der vereinheitlichten grafischen Modellierungssprache UML. Hierbei werden alle relevanten Prozesse einer übergeordneten Aktivität und deren Abhängigkeiten modelliert [Lee13]. In der Softwareentwicklung können dadurch die Abläufe innerhalb einer Software, aber auch Geschäftsabläufe in verschiedenen Detaillierungsgraden dargestellt werden [BPK15]. Abbildung 5.1 zeigt das Aktivitätsdiagramm zur Durchführung einer Optimierung aus Sicht des Anwenders unter Verwendung des zu entwickelnden Optimierungssystems. Beim Durchlaufen der Prozesselemente sind zu jedem Zeitpunkt Iterationen durch das Wiederholen vorangegangener Prozesselemente möglich. Aufgrund der Übersichtlichkeit sind diese Iterationen in der Abbildung nicht dargestellt.

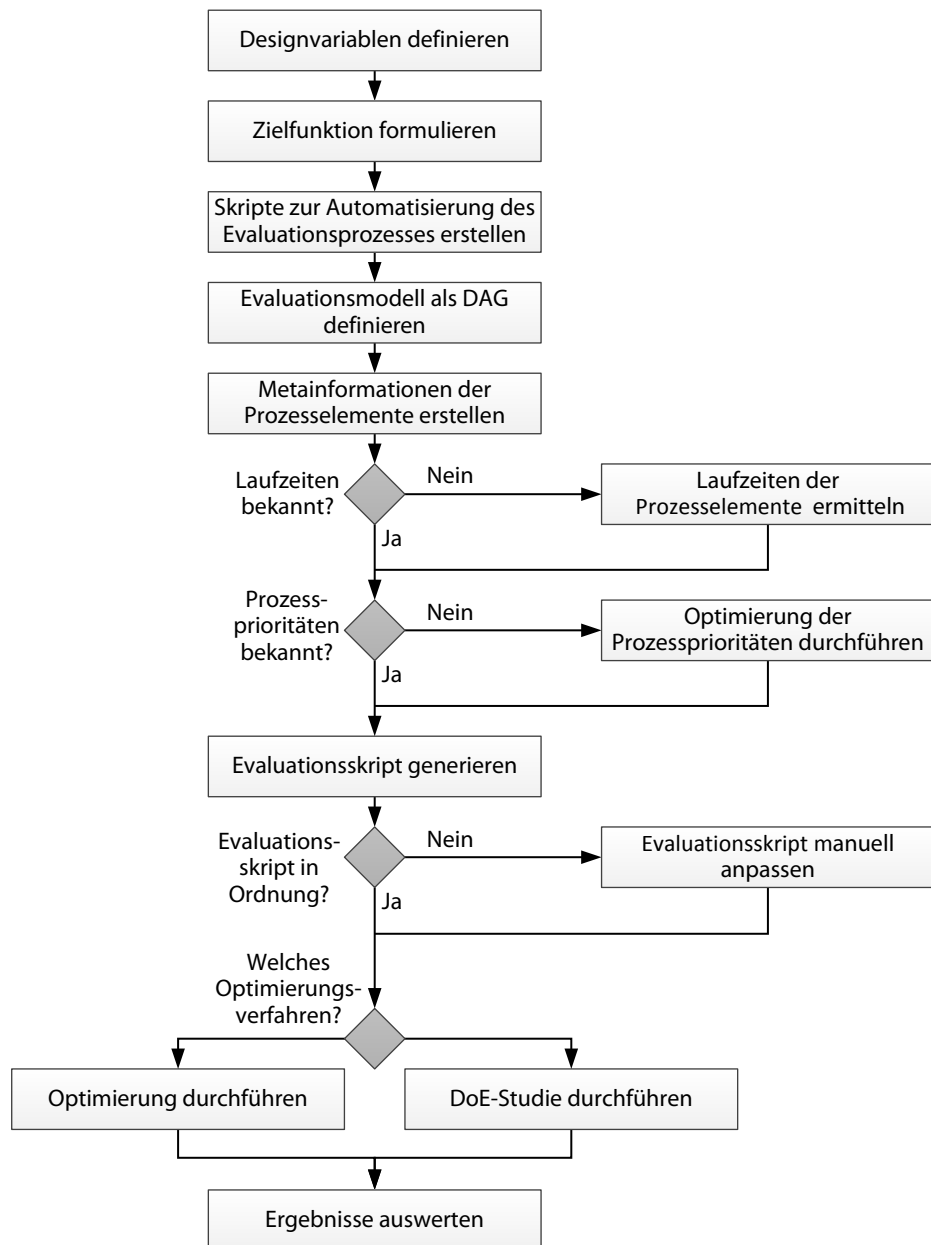


Abbildung 5.1: Aktivitätsdiagramm zur Durchführung einer Optimierung

Jedes Prozesselement des Aktivitätsdiagramms stellt eine Funktion des Optimierungssystems dar, welche bei der Entwicklung realisiert wird. Ausgehend von einer Optimierungsaufgabe erfolgt durch den Anwender zunächst die Definition der Designvariablen und die Formulierung der Zielfunktion. Anschließend werden die Skripte zur Automatisierung des Evaluationsprozesses erstellt. An dieser Stelle findet bereits die Modelldekomposition des Evaluationsmodells statt (s. Abschnitt 4.1). Für jedes Prozesselement des Evaluationsprozesses wird ein separates Automatisierungsskript verwendet. Somit können die Prozesselemente variabel gemäß ihrer benötigten Ressourcen in dem Cluster bearbeitet werden. Zudem lassen sich auf diese Weise auch Vorlagen und Bibliotheken oft verwendeter Prozesselemente erstellen (z. B. die Aktualisierung eines CAD-Modells in CATIA V5 mit anschließendem Export als step-Datei).

Sind die Skripte erstellt, wird das Evaluationsmodell innerhalb des Frameworks als Gerichteter Azyklischer Graph (Directed Acyclic Graph, DAG) definiert. Dieser bildet die

grafische Repräsentation des Evaluationsprozesses und der Abhängigkeiten der Prozesselemente (s. Abschnitt 4.3). Anschließend erfolgt die Eingabe der Metainformationen der Prozesselemente. Hierzu zählen die benötigten Input-Dateien sowie die zur Bearbeitung benötigten Ressourcen. Weiterhin wird an dieser Stelle festgelegt, ob ein Prozesselement auf dem lokalen Rechner oder innerhalb des Clusters bearbeitet werden soll. Entscheidend hierzu sind ebenfalls die benötigten Ressourcen, der numerische Aufwand und die Laufzeit des Prozesselementes. Die Wahl der Parallelisierung über die Metainformationen bildet die Grundlage zur Realisierung des Concurrent und Simultaneous Optimization-Ansatzes, wodurch sich das Framework von konventionellen Systemen unterscheidet. Empfehlungen zur Modelldekomposition und Auswahl der Parallelisierungsmethode werden in Abschnitt 5.6 gegeben.

Zur Realisierung der prioritätenbasierten Bearbeitung der Prozesselemente in dem Cluster (s. Abschnitt 4.5) werden die Laufzeiten der Prozesselemente benötigt. Sind diese dem Anwender bekannt, müssen sie in das System eingegeben werden. Andernfalls werden die Laufzeiten durch das Ausführen des jeweiligen Prozesselementes vom System ermittelt. Analog wird bei der Ermittlung der Prozessprioritäten verfahren. Sind diese bereits bekannt, müssen sie vom Anwender eingegeben werden. Alternativ werden die Prozessprioritäten durch das System ermittelt. Hierzu findet eine Optimierung der Prioritäten statt, indem der zur Verfügung stehende Cluster ausgelesen und die Verteilung der Prozesselemente auf die Ressourcen des Clusters simuliert wird. Das Ziel der Optimierung ist die Minimierung der Laufzeit des gesamten Evaluationsprozesses.

Auf Basis der festgelegten Daten wird anschließend das Evaluationskript generiert. Nach der Generierung kann das Evaluationskript auch manuell durch den Anwender angepasst werden. Im Anschluss an die Auswahl des Optimierungsverfahrens erfolgt die Durchführung der Optimierung bzw. der DoE-Studie und die Auswertung der Ergebnisse durch den Anwender.

Basierend auf den Anforderungen und dem Aktivitätsdiagramm wird das Konzept für das zu entwickelnde Framework erstellt. Dieses beinhaltet die Festlegung der zu verwendenden Systeme sowie deren Aufgaben und Schnittstellen. Die Aufgaben und Interaktionen der in das Framework integrierten Systeme sind in Abbildung 5.2 dargestellt. Die Interaktionen zwischen den Systemen werden durch die Linien repräsentiert.

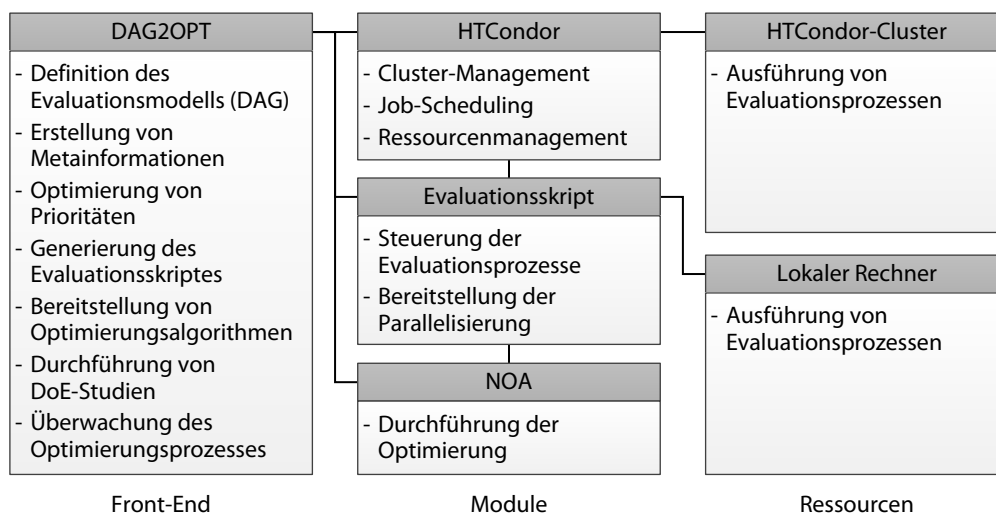


Abbildung 5.2: Aufgaben und Interaktionen der in das Framework integrierten Systeme

Die Schnittstelle zum Anwender bildet das Optimierungssystem DAG2OPT. Bis auf das Erstellen der Skripte zur Automatisierung der Evaluationsprozesse können alle Aktivitäten zur Durchführung einer Optimierung über eine grafische Benutzeroberfläche durchgeführt werden. DAG2OPT dient somit als Front-End des gesamten Frameworks und realisiert die Generierung des Evaluationssskriptes und die Kommunikation zu den Modulen. Basierend auf den Metainformationen der Prozesselemente enthält das Evaluationssskript die vollständige Logik zur Steuerung der Evaluationsprozesse und zur Bereitstellung der flexiblen Parallelisierung der Prozesselemente. Diese werden gemäß ihrer Metainformationen an den HTCCondor-Cluster übermittelt oder auf dem lokalen Rechner ausgeführt, auf dem auch DAG2OPT und die Module ausgeführt werden. Da das Evaluationssskript die vollständige Logik zur Steuerung der Evaluationsprozesse enthält, kann es nach der Generierung auch in externe übergeordnete Prozesse integriert werden. Zur Durchführung einer Optimierung wird das Optimierungssystem NOA verwendet [JC04], welches das Evaluationssskript startet, das Ergebnis der Evaluationen, den Zielfunktionswert, interpretiert und daraus den Fitnesswert ermittelt. Die Steuerung von NOA erfolgt vollständig durch DAG2OPT.

Das Aktivitätsdiagramm und das Konzept bilden die Grundlage des weiteren Vorgehens zur Entwicklung des Frameworks. Dies beinhaltet die Integration und Konfiguration des HTCCondor-Clusters sowie die Entwicklung des Optimierungssystems DAG2OPT. Anhand des Aktivitätsdiagramms können während der Entwicklung kontinuierlich die Relevanz und der Mehrwert für den Anwender validiert werden.

5.3 Clusterintegration und -konfiguration

Die Integration des HTCCondor-Clusters in das zu entwickelnde Framework zur Parallelisierung von Optimierungsverfahren stellt einen wesentlichen Bestandteil der prototypischen Umsetzung dar. HTCCondor stellt dazu eine sehr umfangreiche und vielseitig einsetzbare HTC-Umgebung bereit. Innerhalb des Frameworks soll die Kommunikation zwischen DAG2OPT und dem Cluster selbstständig über HTCCondor erfolgen. Daher wird der Cluster hinsichtlich der Nutzung zur Bearbeitung von Optimierungsaufgaben konfiguriert und die relevanten Schnittstellen spezifiziert, welche durch DAG2OPT verwendet werden.

In den folgenden Abschnitten werden die in Abschnitt 5.1 erarbeiteten Anforderungen zur Integration und Verwendung des Clusters umgesetzt. Für eine detailliertere Beschreibung der verwendeten Funktionen wird auf die Dokumentation von HTCCondor verwiesen [Cen15]. Die vollständigen Konfigurations- und Jobdefinitionsdateien sind im Anhang A.8 zu finden.

5.3.1 Definition und Übermittlung einzelner Jobs

HTCCondor basiert auf dem Master-Worker-Prinzip (s. Abschnitt 4.1). Dabei besteht der Cluster mindestens aus drei Hauptkomponenten: dem verwaltenden Rechner (Central Manager), dem übermittelnden Rechner (Submit Node) und den ausführenden Rechnern (Execute Nodes). Der Central Manager bildet die zentrale Steuereinheit des Clusters und muss in jedem Fall vorhanden sein. In diesem Rechner erfolgt die Verwaltung der zu bearbeitenden Berechnungsjobs und der verfügbaren Ressourcen der ausführenden Rechner.

Die Verwaltung der Berechnungsjobs und Ressourcen stellt die Basis für die Zuordnung der Berechnungsjobs zu den jeweiligen ausführenden Rechnern dar. Diese Zuordnung wird als *Matchmaking* bezeichnet [Cen15]. Das Übermitteln eines Jobs an den Cluster erfolgt durch einen übermittelnden Rechner. Je nach Konfiguration des Clusters können verschiedene Rechner übermittelnde Berechtigungen besitzen. Gleiches gilt für die ausführenden Rechner. Diese können je nach Verfügbarkeit und Auslastung variabel dem Cluster hinzugefügt oder entfernt werden, worin eindeutig ein Vorteil des Systems liegt. Die Hauptkomponenten eines HTCondor-Clusters zur Jobübermittlung und -verteilung sind in Abbildung 5.3 dargestellt.

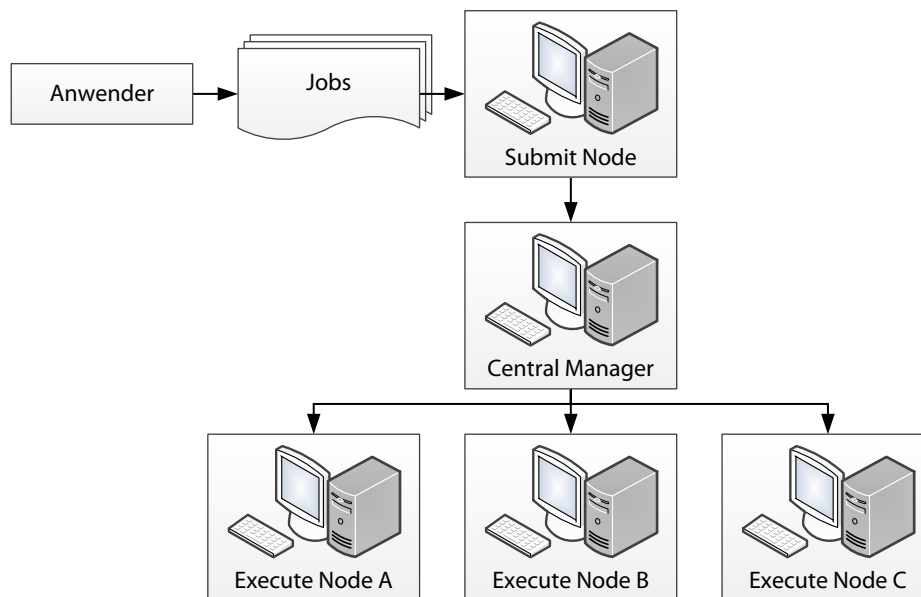


Abbildung 5.3: Hauptkomponenten eines HTCondor-Clusters, nach [Lim10]

Innerhalb des Clusters kann je nach Konfiguration jeder Rechner unterschiedliche Rollen einnehmen. So kann ein Rechner gleichzeitig Submit-Rechner und auch Execute-Rechner sein. Die Rolle des Central Managers bleibt in der Regel fest mit einem Rechner verknüpft, wobei dieser je nach Konfiguration ebenfalls Jobs übermitteln oder Jobs ausführen kann. Durch diese Modularität ist der Cluster sehr flexibel anpassbar und skalierbar. Die jeweilige Rolle eines Rechners wird über die im Hintergrund laufenden Prozesse (Dienste bzw. Daemons) realisiert. Die aufgrund der unterschiedlichen Rollen auf den jeweiligen Rechnern ausgeführten Dienste sind in Abbildung 5.4 dargestellt. Dabei werden die in HTCondor verwendeten Bezeichnungen genutzt.

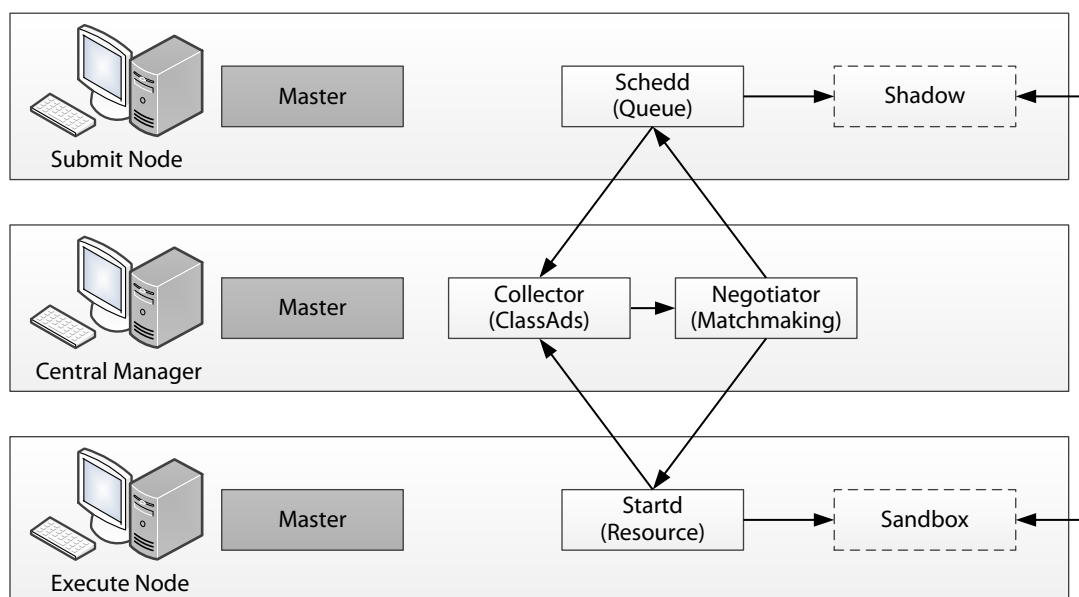


Abbildung 5.4: Wesentliche Dienste der Hauptkomponenten eines HTCondor-Clusters, nach [HK04], [Lim10]

Auf jedem Rechner des HTCondor-Clusters läuft unabhängig von dessen Rolle ein Master-Dienst (*condor_master*), welcher je nach Konfiguration die rollenspezifischen Dienste des Rechners steuert. Die Rolle des Central Managers wird durch die Dienste Collector (*condor_collector*) und Negotiator (*condor_negotiator*) realisiert. Der Submit-Rechner beinhaltet den Scheduler, welcher durch den Schedd-Dienst (*condor_schedd*) realisiert wird. Die Bearbeitung von Berechnungsjobs auf dem Execute-Rechner wird durch den Startd-Dienst (*condor_startd*) ausgeführt.

Der Collector-Dienst sammelt alle relevanten Informationen über die Submit- und die Execute-Rechner, die Klassenattribute (ClassAds). Diese beinhalten Informationen über den Status der Dienste, die Ressourcen und die Ressourcenanfragen durch Jobs. Die Auswertung der Klassenattribute erfolgt durch den Negotiator-Dienst. Dieser vergleicht die Attribute miteinander und bildet zusammenpassende Paare aus Jobanfragen und Ressourcen (Matchmaking). Wird ein Job durch den Anwender von einem Submit-Rechner aus übermittelt, wird der Job zunächst durch den Schedd-Dienst in eine interne Warteschlange (Queue) eingeordnet. Der Job erhält eine eindeutige Beschreibung, welche dem Collector-Dienst übermittelt wird. Wird durch den Negotiator ein passender Execute-Rechner gefunden, wird dies an die jeweiligen Submit- und Execute-Rechner kommuniziert, woraufhin die Schnittstellendienste Shadow und Sandbox gestartet werden, welche den Austausch der relevanten Daten realisieren. Nach Fertigstellung des Jobs wird ein Statusbericht an den Submit-Rechner übermittelt und beide Dienste werden beendet [HK04], [Lim10].

Berechnungsjobs werden in HTCondor durch Jobdefinitionsdateien definiert, welche an den Scheduler übermittelt werden. Der folgende Quellcode zeigt beispielhaft den Aufbau und den Inhalt einer solchen Datei. Eine vollständige Jobdefinitionsdatei, wie sie in den in dieser Arbeit durchgeführten Fallstudien verwendet wird, ist im Anhang A.8.3 dargestellt.

```

1 # HTCondor Jobdefinitionsdatei
2 universe = vanilla
3
4 executable          = evaluation_1_update_preprocessing.bat
5 transfer_input_files = Scripts, Kurbelarm_links_sim1.sim, Kurbelarm_links_fem1.fem,
6                     Kurbelarm_links.prt, parameter.txt
7 #transfer_output_files =
8
9 log                = evaluation_1_update_preprocessing.log
10 output            = evaluation_1_update_preprocessing.out
11 error             = evaluation_1_update_preprocessing.err
12
13 should_transfer_files = yes
14 when_to_transfer_output = on_exit
15
16 #initialdir        = C:\path
17 #load_profile       = true
18
19 queue

```

Eine Jobdefinitionsdatei muss mindestens die Befehle `universe`, `executable` und `queue` enthalten. Zusätzlich können weitere optionale Informationen festgelegt werden, z. B. für den Dateitransfer, die Dateiausgabe, die Benutzerverwaltung und die Ressourcenauswahl. Die wichtigsten Einträge werden folgend erläutert:

universe: In HTCondor können verschiedene Laufzeitumgebungen, sog. Universen, verwendet werden, in denen die Berechnungsjobs ausgeführt werden. Somit können verschiedene Arten von Anwendungen eingebunden werden. Die Definition des zu verwendenden Universums muss in jeder Jobdefinitionsdatei enthalten sein [Cen15].

Das Vanilla-Universum ist die einfachste der verfügbaren Umgebungen. Programme müssen hier nicht separat verlinkt, sondern können direkt ausgeführt werden, was die Ausführung von Skripten erleichtert (z. B. zur Automatisierung von CAx-Systemen). Die Funktion des Checkpointing wird in dieser Umgebung nicht unterstützt. Diese ermöglicht das Pausieren laufender Jobs und die Übertragung des Zustands eines pausierten Jobs auf einen anderen Rechner, auf dem der pausierte Job dann fertiggestellt wird. Da diese Funktion nicht unterstützt wird, kann ein laufender Job, der pausiert wird, z. B. aufgrund von interaktiver Arbeit an dem ausführenden Rechner, nur auf diesem Rechner direkt fortgesetzt werden. Wird der Job an einen anderen Rechner übermittelt, muss er vollständig neu gestartet werden. Weiterhin können keine Remote-System-Calls verwendet werden, um auf die Eingabe- und Ausgabedateien zuzugreifen. Alternativ können hier ein gemeinsames Dateisystem (z. B. ein Netzlaufwerk) oder der Dateitransfer durch HTCondor verwendet werden [Cen15]. Dies stellt jedoch keine Einschränkung dar.

Im Standard-Universum werden im Gegensatz zum Vanilla-Universum die Funktionen des Checkpointing und Remote-System-Calls unterstützt. Auszuführende Programme müssen in dieser Umgebung jedoch über den Befehl `condor_compile` mit HTCondor verknüpft werden [Lim10].

Aufgrund der einfachen Ausführung von Skripten wird für die in dieser Arbeit verwendeten Berechnungsjobs das Vanilla-Universum gewählt. Der Dateitransfer erfolgt durch HTCondor.

executable: Durch diesen Befehl wird das Programm oder das Skript festgelegt, welches den Berechnungsjob ausführt. Hierbei ist die Verwendung von absoluten und relativen

Pfaden möglich. Das Programm oder das Skript wird zur Bearbeitung in ein temporäres Arbeitsverzeichnis des ausführenden Rechners übertragen (`C:\condor\execute`).

transfer_input_files: Die angegebenen Dateien oder Ordner werden zur Bearbeitung eines Jobs auf den ausführenden Rechner übertragen und in das temporäre Arbeitsverzeichnis kopiert. Soll der Inhalt eines Ordners in das Stammverzeichnis des ausführenden Rechners übertragen werden, muss hinter der Ordnerbezeichnung ein Schrägstrich folgen (z. B. `path/`). Das Übertragen der gesamten Verzeichnisstruktur erfolgt nur, wenn kein Schrägstrich verwendet wird (z. B. `path`) [Cen15].

transfer_output_files: Nach Fertigstellung eines Jobs werden die durch diesen Befehl definierten Dateien wieder auf den Rechner transferiert, von dem der Job übermittelt wurde. Wird der Befehl nicht verwendet, werden alle Ergebnisdateien des Jobs übertragen. Durch diese Funktion können die Ergebnisdateien eines Berechnungsjobs gefiltert werden, wenn nur bestimmte Dateien benötigt werden.

log, output, error: In der Log-Datei werden alle Informationen über die Ausführung eines Jobs festgehalten. In der Output-Datei wird die gesamte Ausgabe des ausgeführten Programmes gespeichert. Die Error-Datei beinhaltet im Falle eines Fehlers bei der Jobbearbeitung auftretenden Fehlermeldungen.

should_transfer_files: Dieser Befehl definiert den Transfer relevanter Dateien des Jobs. Der Dateitransfer wird über die Optionen `yes` und `no` aktiviert bzw. deaktiviert. Wird die Option `if_needed` verwendet, findet der Dateitransfer nur statt, wenn übermittelnder und ausführender Rechner nicht auf das gleiche Dateisystem (z. B. Netzlaufwerk) zugreifen.

when_to_transfer_output: Durch diesen Befehl wird der Zeitpunkt definiert, wann die Ausgabedaten an den übermittelnden Rechner transferiert werden. Die Option `on_exit` legt fest, dass dies erst nach der vollständigen Beendigung des Jobs erfolgen soll.

initialdir: Mit diesem Befehl wird das Arbeitsverzeichnis auf dem ausführenden Rechner festgelegt.

load_profile: Durch diesen Befehl kann das aktuelle Benutzerprofil von dem übertragenden Rechner auf den ausführenden Rechner übertragen werden. Dies ist insbesondere notwendig, wenn die Bearbeitung eines Jobs Benutzereinstellungen oder -daten erfordert (z. B. bei der Verwendung eines Visual Basic-Skriptes in NX).

queue: Dieser Befehl steht in der Regel am Ende der Jobdefinitionsdatei und ermöglicht die Integration des Jobs in die Warteschlange des Schedulers. Wird hinter dem Befehl eine ganze Zahl verwendet, so bestimmt diese Zahl die Anzahl verschiedener Instanzen des Jobs in der Warteschlange. Bei der Verwendung in Evaluationsprozessen von Optimierungsproblemen, ist die einmalige Ausführung eines Jobs ausreichend, da der Aufruf der Prozesse durch den Optimierer gesteuert wird.

Durch die Übermittlung der Jobdefinitionsdatei an den Scheduler wird der Job der Warteschlange hinzugefügt. Die Übermittlung erfolgt durch den folgenden Befehl:

```
1 # Übermitteln der Jobdefinitionsdatei
2 condor_submit <Jobdefinitionsdatei>
```

Bei der erstmaligen Verwendung des Systems muss der Anwender registriert werden. Dies erfolgt in der Kommandokonsole des übermittelnden Rechners durch die folgende Anweisung:

```
1 # Uebermitteln der Jobdefinitionsdatei  
2 condor_store_cred add -p <Passwort>
```

Das Passwort muss hierbei identisch mit dem Benutzerkennwort des Betriebssystems bzw. mit dem jeweiligen Kennwort der Domäne sein. Die Benutzerregistrierung muss an jedem Rechner durchgeführt werden, von dem ein Job übermittelt wird [Cen15].

5.3.2 Definition und Übermittlung zusammenhängender Jobs

Aufgrund der Dekomposition des Evaluationsmodells (s. Abschnitt 4.1) entstehen mehrere zusammenhängende und voneinander abhängige Prozesselemente, welche in einem Workflow zusammengefasst werden (s. Abschnitt 4.3). Workflows stellen eine Erweiterung einfacher Jobmodelle dar und sind insbesondere für den Einsatz im Cluster- und Grid-Computing geeignet. Die Modellierung und Beschreibung eines Workflows erfolgt dabei als gerichteter azyklischer Graph (Directed Acyclic Graph, DAG), wobei die einzelnen Prozesselemente bzw. Jobs als Knoten abgebildet werden. Die Abhängigkeiten und somit die Reihenfolge der Knoten werden durch Linien dargestellt [CKR⁺07], [BBKS15]. Abbildung 5.5 zeigt eine einfache Beschreibung eines Workflows als DAG.

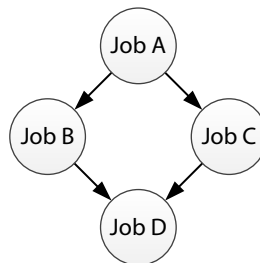


Abbildung 5.5: DAG-Beschreibung eines Workflows, nach [CKR⁺07]

Im Gegensatz zur DAG-Definition als Task-Präzedenz-Graph, welche rein statischer Natur ist, können mit Workflow-Beschreibungen auch komplexe und dynamische Strukturen wie Schleifen, bedingte Verzweigungen oder Verschachtelungen beschrieben werden [BBKS15].

Innerhalb von HTCondor wird ein DAG durch das DAGMan-Modul (Directed Acyclic Graph Manager) bearbeitet. Da in HTCondor selbst keine Abhängigkeiten der Jobs untereinander berücksichtigt werden, operiert das DAGMan-Modul auf einer übergeordneten Ebene als Workflow-Management-System bzw. Meta-Scheduler und verwaltet dort die Abhängigkeiten der Jobs [CKR⁺07]. Zur Berücksichtigung der Abhängigkeiten wird eine separate Eingabedatei verwendet, in der die deklarative Beschreibung der Jobs und deren Zusammenhänge über spezielle Parent-Child-Anweisungen erfolgt. Die Jobs werden durch das DAGMan-Modul in der richtigen Reihenfolge an den Scheduler-Dienst übergeben [BBKS15].

Eine DAGMan-Eingabedatei besteht aus den folgenden Elementen:

- **JOB**: Deklaration der Jobdefinitionsdateien
- **PARENT/CHILD**: Definition der Abhängigkeiten
- **SCRIPT PRE/POST**: Pre- und Postskripte
- **RETRY**: Wiederholungsanweisung

Zunächst werden die Jobdefinitionsdateien deklariert (**JOB**). Dabei wird jede verwendete Datei mit einer Bezeichnung versehen und der Speicherort der Datei festgelegt. Mit Hilfe der Bezeichnungen werden die Abhängigkeiten definiert. Ein Eltern-Knoten (**PARENT**) kann dabei beliebig viele Kinder-Knoten (**CHILD**) besitzen. Ebenso kann ein Kinder-Knoten beliebig viele Eltern-Knoten haben. Die Verwendung der Pre- und Postskripte sowie der Wiederholungsanweisung ist nicht zwingend notwendig, bietet aber zusätzliche Anwendungsmöglichkeiten für die Fehlerbehandlung (s. Abschnitt 5.3.5). Der folgende Quellcode beinhaltet die Deklaration der Jobdefinitionsdateien und die Definition der Abhängigkeiten für den in Abbildung 5.5 dargestellten DAG:

```
1 # Jobdefinitionsdateien
2 JOB A C:\A.job
3 JOB B C:\B.job
4 JOB C C:\C.job
5 JOB D C:\D.job
6
7 # Abhaengigkeiten
8 PARENT A CHILD B C
9 PARENT B C CHILD D
```

Die Übermittlung von DAGMan-Eingabedateien an den Scheduler erfolgt über den folgenden Befehl:

```
1 # Uebermitteln der DAGMan-Eingabedatei
2 condor_submit_dag <DAGMan-Eingabedatei>
```

5.3.3 Monitoring von Ressourcen und Jobs

Während der Arbeit mit HTCondor ist das Monitoring und Editieren laufender Jobs eine zentrale Aufgabe. Das Monitoring beinhaltet zwei Komponenten: die Ressourcen des Clusters (Rechner) und die Warteschlange der Jobs. Der Status des Clusters wird über den folgenden Befehl in der Kommandokonsole ausgelesen:

```
1 # Status des Rechner-Pools
2 condor_status
```

Neben der Auflistung der im Cluster vorhanden Rechner können zusätzliche Optionen zum Filtern der angezeigten Rechner verwendet werden. So können die zurzeit verfügbaren oder belegten Rechner sowie die Rechner, welche sich im Standby-Modus befinden, angezeigt werden [Cen15]. Nach Ausführen des Befehls erscheint in der Kommandokonsole folgende Übersicht:

Name	OpSys	Arch	State	Activity	LoadAv	Mem	ActvtyTime
PSG31.mb.uni-magde	WINDOWS	X86_64	Claimed	Busy	1.000	16341	0+00:00:30
PSG32.mb.uni-magde	WINDOWS	X86_64	Unclaimed	Idle	0.070	16341	0+01:03:50
PSG33.mb.uni-magde	WINDOWS	X86_64	Owner	Idle	0.130	16341	0+00:24:10
	Total	Owner	Claimed	Unclaimed	Matched	Preempting	Backfill
X86_64/WINDOWS	3	1	1	1	0	0	0
Total	3	1	1	1	0	0	0

In der Konsole werden die Rechnerbezeichnung (**Name**), das Betriebssystem (**OpSys**), die Rechnerarchitektur (**Arch**) sowie der Status (**State**) und die Aktivität (**Activity**) angezeigt. Weiterhin werden die Auslastung (**LoadAv**) und der vorhandene Speicher eines Rechners (**Mem**) sowie die Zeit, seit der der Rechner die aktuelle Aktivität durchführt (**ActvtyTime**), dargestellt. Die wichtigsten Zustandsgrößen für die Überwachung von Jobs sind der Status und die Aktivität eines Rechners. Ein mit dem Pool verbundener Rechner kann hierbei folgende Zustände aufweisen:

Unclaimed: Der Rechner ist verfügbar und bereit, einen Job anzunehmen.

Owner: Der Rechner wird gerade von einem Anwender zur interaktiven Arbeit verwendet.

Claimed: Auf dem Rechner wird ein Job ausgeführt.

Matched: Der Rechner erfüllt die Bedingungen zum Ausführen eines Jobs und wird diesen in Kürze ausführen.

Preempting: Der auf dem Rechner ausgeführte Job wird beendet und von diesem Rechner entfernt.

Ergänzend zum Status eines Rechners wird die Aktivität des Rechners ausgegeben. Ein Rechner kann eine der folgenden Aktivitäten ausführen:

Idle: Der Rechner führt keine Aktivität aus und steht zur Verfügung.

Busy: Auf dem Rechner wird ein Job ausgeführt.

Suspended: Der laufende Job wird pausiert z. B. aufgrund der interaktiven Arbeit durch einen Anwender.

Killing: Der Job wird auf diesem Rechner abgebrochen und entfernt.

Benchmarking: Auf diesem Rechner wird ein Benchmark-Test durchgeführt, um die Leistung des Computers zu bestimmen.

Die zweite Komponente des Monitorings ist die Warteschlange der Jobs, welche über den folgenden Befehl ausgegeben wird:

```
1 # Status der Warteschlange
2 condor_q
```

Auch hier können zusätzlich verschiedene Optionen wie das Filtern nach bestimmten Anwendern oder eine umfassende Jobanalyse verwendet werden [Cen15]. Nach Ausführen des Befehls wird in der Kommandokonsole folgende Übersicht dargestellt:


```

1 -- Schedd: PSG56.mb.uni-magdeburg.de : <141.44.133.56:46718?...
2 ID      OWNER/NODENAME  SUBMITTED   RUN_TIME ST PRI SIZE CMD
3 1.0     awuensch            1/29 09:51  0+00:01:03 R  1  0.0  evaluation_1_updat
4 2.0     awuensch            1/29 09:51  0+00:00:00 I  0  0.0  evaluation_2_solvi
5
6 2 jobs; 0 completed, 0 removed, 1 idle, 1 running, 0 held, 0 suspended

```

Jeder an die Warteschlange übermittelte Job erhält eine eindeutige ID zur Identifizierung. Diese ID beinhaltet die Clusterkennung (ClusterID) und die aktuelle Prozesskennung (ProcessID, Standardwert 0). Die ClusterID wird ausgehend von einem Wert von 1 fortlaufend jedem Job zugewiesen. Die ProcessID spiegelt die Instanz eines Jobs wider, welche in der Jobdefinitionsdatei durch den Befehl `queue` festgelegt wurde. Weiterhin werden der Benutzername des übermittelnden Anwenders (`OWNER`), die aktuelle Laufzeit (`RUN_TIME`), der Status (`ST`), die Priorität (`PRI`), die Größe (`SIZE`) und das innerhalb des Jobs ausgeführte Programm bzw. Skript (`CMD`) angezeigt.

Der Status eines Jobs stellt hierbei die wichtigste Information zur Überwachung dar. Ein Job kann einen der folgenden Zustände einnehmen:

I - Idle: Der Job befindet sich in der Warteschlange und ist bereit zur Ausführung.

R - Running: Der Job wird ausgeführt.

H - Hold: Der laufende Job wurde angehalten und wird nicht fortgesetzt, bis der Haltestatus beendet wurde. Dieser Status ist meist ein Indiz für eine fehlerhafte Bearbeitung des Jobs.

S - Suspended: Der Job wurde pausiert z. B. weil der Rechner, auf dem der Job ausgeführt wurde, interaktiv verwendet wird. Steht dieser Rechner wieder Verfügung, wird der Job fortgesetzt, andernfalls wird der Job einem anderen Rechner zugeordnet.

C - Completed: Der Job wurde erfolgreich bearbeitet.

X - Removed: Der Job wird aus der Warteschlange entfernt.

Die Steuerung von HTCCondor erfolgt ausschließlich durch Befehle in der Kommandokonsole. Somit ist das System sehr flexibel und vielseitig einsetzbar und kann einfach in Skripte und Programmabfolgen implementiert werden, da die Befehle zur Steuerung gleichzeitig die Anwendungsprogrammierschnittstelle (Application Programming Interface, API) darstellen.

Die interaktive Arbeit, insbesondere das Monitoring und die Editierung laufender Jobs, wird durch die fehlende grafische Benutzungsoberfläche (GUI) jedoch deutlich erschwert. Zudem erhält der Anwender in der Kommandokonsole nur eine statische Ausgabe der Ressourcen und der Warteschlange. Eine dynamische Anzeige, in der selbstständig Veränderungen dargestellt werden, ist nicht möglich. Diese Nachteile in der Anwendung von HTCCondor werden durch das in der Programmiersprache Python (Version 3.4.3) [Pyt16] entwickelte Programm *Condor GUI* eliminiert [WD16].

Das Programm beinhaltet folgende Funktionen zur interaktiven Arbeit mit HTCCondor:

- Bereitstellung von Statusinformationen des Clusters und der Warteschlange mit verschiedenen Anzeigefiltern
- Anzeige der definierten Klassenattribute
- Erstellung und Editierung von Jobdefinitions- und DAGMan-Eingabedateien
- Übermittlung von Jobdefinitions- und DAGMan-Eingabedateien
- Entfernung laufender Jobs
- Editierung der Prioritäten einzelner Jobs
- Kommandozeile zur Eingabe anwenderspezifischer Befehle
- Texteditor zur Bearbeitung der HTCCondor-Konfigurationsdatei inklusive spezifischer Syntaxhervorhebung
- Rekonfiguration und Neustart des Clusters
- Schnittstelle zur Visualisierung von Nutzungsstatistiken des HTCCondorView-Clients

Der wesentliche Bestandteil von Condor GUI ist die Hauptoberfläche, welche in Abbildung 5.6 dargestellt ist. Die Oberfläche ist in zwei Bereiche aufgeteilt, in denen die Statusinformationen des Clusters und die der Warteschlange angezeigt werden. Weiterhin können in den jeweiligen Bereichen verschiedene Anzeigefilter und Optionen verwendet werden.

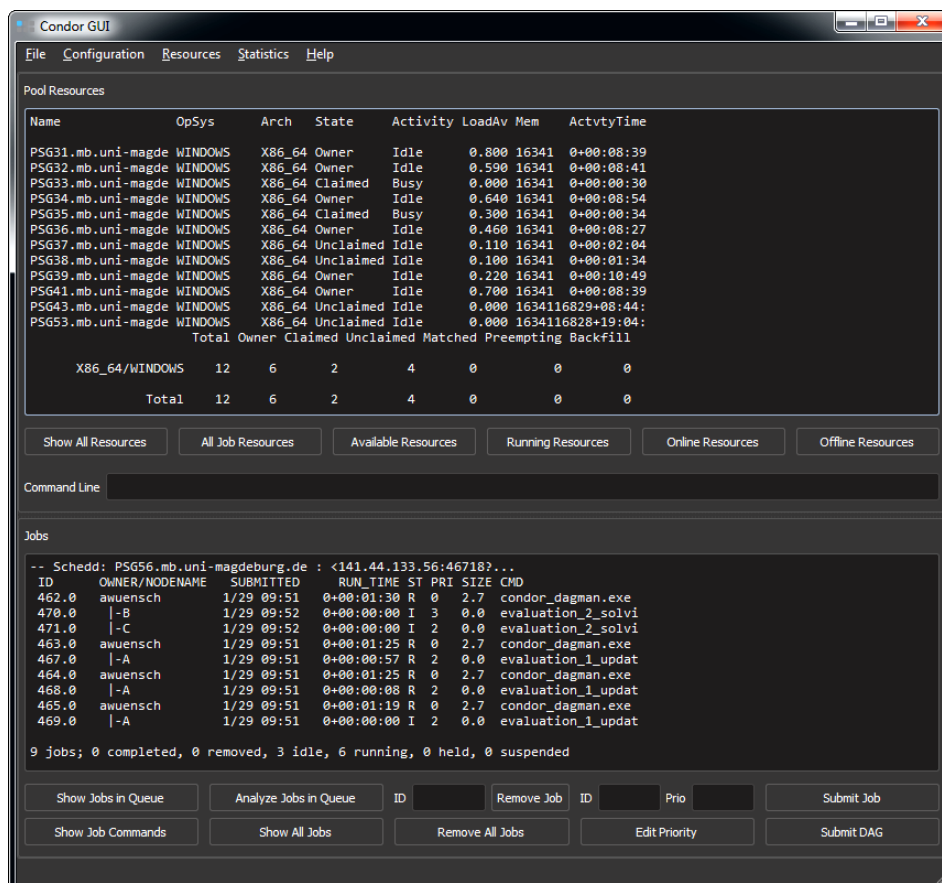


Abbildung 5.6: Hauptoberfläche des Programms Condor GUI

Condor GUI ist modular aufgebaut. Die verschiedenen Module (z. B. die Erstellung und Editierung von Jobdefinitionsdateien) werden aus der Hauptoberfläche heraus aufgerufen. Somit wird die Anpassbarkeit und Erweiterbarkeit des Programms gewährleistet, da einfach vorhandene Module verändert bzw. neue Module hinzugefügt werden können.

5.3.4 Priorisierung von Jobs

Durch die prioritätenbasierte Bearbeitung von Berechnungsjobs kann die Laufzeit einer Optimierung reduziert werden, da Leerlaufzeiten der vorhandenen Ressourcen vermieden und diese effizient genutzt werden (s. Abschnitt 4.5).

Zur Implementierung der prioritätenbasierten Bearbeitung in HTCCondor können zwei Methoden genutzt werden: die Implementierung in der Jobdefinitionsdatei oder in der DAGMan-Eingabedatei. In HTCCondor besitzen alle Jobs standardmäßig die Priorität 0. In der Warteschlange werden die Jobs gleicher Priorität nach dem FIFO-Prinzip verarbeitet.

Über den Befehl `priority` wird in der Jobdefinitionsdatei die Priorität eines einzelnen Jobs definiert, wie im folgenden Quellcode dargestellt ist:

```
1 # Jobdefinitionsdatei
2 priority = 3
```

Die Priorität eines Jobs kann Werte von -2147483648 bis $+2147483647$ annehmen. Dabei wird stets eine höhere Priorität einer niedrigeren vorgezogen.

Die Prioritätsdefinition in der Jobdefinitionsdatei ist in HTCCondor nur für einzelne Jobs möglich und kann zur Festlegung der Bearbeitungsreihenfolge voneinander unabhängiger Jobs in der Warteschlange genutzt werden. Da Evaluationsprozesse von Optimierungsaufgaben in den meisten Fällen aus mehreren Prozesselementen bestehen, welche durch einen DAG abgebildet werden, wird die Implementierung in der DAGMan-Eingabedatei genutzt. Dazu werden den Jobs nach der Deklaration und der Definition der Abhängigkeiten die Prioritätswerte über den Befehl `PRIORITY` zugewiesen. Dies ist im folgenden Quellcode dargestellt:

```
1 # DAGMan-Eingabedatei
2 JOB A evaluation_1_update_preprocessing.job
3 JOB B evaluation_2_solving_eigenwerte.job
4 JOB C evaluation_2_solving_linear_static.job
5
6 PARENT A CHILD B C
7
8 PRIORITY A 1
9 PRIORITY B 3
10 PRIORITY C 2
```

Die in diesem Abschnitt dargestellten Quellcodes stellen nur Ausschnitte einer Jobdefinitionsdatei bzw. einer DAGMan-Eingabedatei dar. Der vollständige Inhalt der Dateien ist im Anhang A.8.3 und Anhang A.8.4 zu finden.

Werden zusammenhängende Jobs als DAG übermittelt, werden die Jobprioritäten der einzelnen Jobs in den jeweiligen Jobdefinitionsdateien ignoriert und die Prioritäten der

DAGMan-Eingabedatei verwendet. Wird keine Priorität in der DAGMan-Eingabedatei verwendet, erhält der Job eine Priorität von 0.

Bei der Prioritätenzuweisung in der DAGMan-Eingabedatei findet zusätzlich eine Vererbung der Prioritäten statt, wenn der Prioritätswert eines Jobs geringer ist als der seines benötigten Vorgängers. Wird also z. B. Job A eine Priorität von 4 zugewiesen, wird dieser Wert in dem dargestellten Beispiel an Job B und C vererbt, da diese Prioritätswerte kleiner als 4 sind. Die nachfolgenden Jobs eines DAG erhalten also mindestens die Priorität ihres benötigten Vorgängers [Cen15]. Dadurch wird verhindert, dass bereits gestartete DAG durch Jobs höherer Priorität zurückgestellt und nicht fortgesetzt werden. Das System ist immer bestrebt, bereits gestartete DAG zu beenden, bevor neue Jobs gestartet werden.

Der Prioritätswert eines Jobs ist nach dessen Übermittlung weiter editierbar und kann durch den folgenden Quellcode verändert werden:

```
1 # Editierung des Prioritaetswertes
2 condor_prio -p <Wert> <ID>
```

Der Befehl `condor_prio` wird dazu mit dem neuen Prioritätswert und der ID des Jobs als Argumenten ausgeführt. Zur schnellen und einfachen Zuweisung neuer Prioritätswerte ist dieser Befehl in dem Programm Condor GUI implementiert (s. Abschnitt 5.3.3).

5.3.5 Fehlerbehandlung bei Lizenzfehlern

Treten bei der Bearbeitung von Berechnungsjobs unerwartete Fehler auf, ist es wichtig, dass das System selbstständig darauf reagieren und eine Fehlerbehandlung einleiten kann, was vor allem für die Behandlung bei Lizenzfehlern sinnvoll ist. Bei Auftreten eines solchen Fehlers durch das Fehlen einer Lizenz soll der Fehler erkannt und der Berechnungsjob nach einer festgelegten Wartezeit wieder an den Scheduler übermittelt werden. Die Behandlung bei Lizenzfehlern ist innerhalb von HTCondor mittels zwei Methoden möglich: durch Verwendung vordefinierter Fehlercodes und durch Verwendung von Pre- und Postskripten in Kombination mit der Wiederholungsanweisung in der DAGMan-Eingabedatei.

Die erste Methode basiert auf Fehlercodes, welche das Auftreten von Lizenzfehlern dokumentieren. Bei der Beendigung eines Jobs werden die Ausgabewerte und aufgetretene Fehler in die Output-Datei bzw. in die Error-Datei geschrieben. Enthält die Error-Datei einen lizenzfehlerspezifischen Fehlercode, kann durch den Befehl `on_exit_remove` festgelegt werden, wann der Job die Warteschlange verlässt. Dies erfolgt über den folgenden Quellcode in der Jobdefinitionsdatei:

```
1 # Jobdefinitionsdatei
2 on_exit_remove = (ExitBySignal == False) && (ExitCode == 0)
```

Die Implementierung dieser Methode ist sehr einfach. Bei der Bearbeitung von Optimierungsaufgaben kommen in der Regel verschiedene CAx-Systeme zum Einsatz, in denen Fehler auf unterschiedliche Weise codiert werden. Zudem erfolgt die Fehlercodierung bei den meisten Systemen nicht fehlerspezifisch. Anhand des Fehlercodes kann die Art des Fehlers somit nicht identifiziert werden. Diese Methode kann also lediglich bei selbst entwickelten Programmen verwendet werden und wird daher als nicht zielführend bewertet [Dzi15].

Die zweite Methode der Lizenzfehlerbehandlung in HTCondor sieht die Verwendung von Pre- und Postskripten in Verbindung mit der Wiederholungsanweisung in der DAGMan-Eingabedatei vor. Abbildung 5.7 zeigt den um die vorgestellte Methode erweiterten DAG aus Abbildung 5.5.

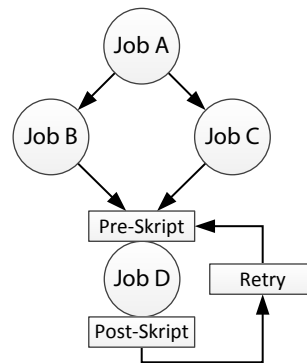


Abbildung 5.7: Verwendung von Pre- und Postskripten in einem DAG, nach [TTL05], [CKR⁺07]

Bevor Job D ausgeführt wird, wird auf dem Rechner, von welchem aus dieser Job übermittelt wurde, das Pre-Skript ausgeführt. Ein Pre-Skript kann hierbei z. B. die Konvertierung von Eingabedateien übernehmen oder eine für den Job notwendige Umgebung generieren. Nach Beendigung des Jobs wird das Post-Skript gestartet, ebenfalls auf dem Rechner, von welchem aus der Job übermittelt wurde. Wird in diesem Skript auf Basis der Ausgabedateien ein spezieller Fehler (z. B. ein Lizenzfehler) erkannt, erfolgt die Wiederholungsanweisung (Retry).

Diese Methode zur Lizenzfehlerbehandlung wird durch die im folgenden Quellcode dargestellte DAGMan-Eingabedatei realisiert:

```

1  # Jobdefinitionsdateien
2  JOB A C:\A.job
3  JOB B C:\B.job
4  JOB C C:\C.job
5  JOB D C:\D.job
6
7  # Abhaengigkeiten
8  PARENT A CHILD B C
9  PARENT B C CHILD D
10
11 # Fehlerbehandlung
12 SCRIPT PRE D C:\pre_script.bat
13 SCRIPT POST D C:\post_script.bat \$$DAG_STATUS
14 RETRY D 5

```

Das Post-Skript wird mit dem in HTCondor vordefinierten Makro `DAG_STATUS` als Argument aufgerufen. Dieses Makro liefert den Wert 0, wenn der Job fehlerfrei bearbeitet wurde. Andernfalls wird ein Wert ungleich 0 übergeben. In diesem Fall erfolgt die Identifizierung des Fehlers durch das Postskript. Hierzu kann z. B. die Log-Datei des Jobs nach der Art des Fehlers untersucht werden.

Wird ein Lizenzfehler detektiert, so wird ein Wert von 1 zurückgegeben und die Wiederholungsanweisung ausgeführt. Durch den Befehl `RETRY` wird festgelegt, wie oft der Job

bei einem Fehler wiederholt werden soll. Bei der Identifizierung eines Lizenzfehlers ist es sinnvoll, dass der Job nicht sofort wieder an den Scheduler übermittelt und bearbeitet wird, da der Lizenzfehler sonst mit hoher Wahrscheinlichkeit wieder auftritt. Daher ist die Implementierung einer Wartezeit sinnvoll. Nach dieser Wartezeit kann die benötigte Lizenz wieder zur Verfügung stehen, da z. B. keine Nutzung mehr durch einen anderen Anwender erfolgt oder die Funktion des Lizenzservers überprüft wurde. Wird durch das Post-Skript ein Wert von 0 zurückgegeben, wird die Wiederholungsanweisung nicht ausgeführt [Dzi15]. Ein mögliches Post-Skript zur Fehlerbehandlung bei Lizenzfehlern ist im Anhang A.8.5 dargestellt.

5.3.6 Definition und Verwaltung von Ressourcen

Innerhalb von HTCondor werden in erster Linie die ausführenden Rechner als Ressourcen angesehen. Ausgehend von einer flexiblen Bereitstellung der zur Verfügung stehenden Hardware werden in diesem Abschnitt zusätzlich die installierte Software sowie die Anzahl der verfügbaren Lizenzen als Ressourcen berücksichtigt.

Die Rechner des Clusters sollen neben der Bereitstellung von Rechenkapazitäten zur Bearbeitung von Optimierungsaufgaben auch weiterhin zur interaktiven Arbeit zur Verfügung stehen. Ein Rechner, welcher interaktiv genutzt wird, soll in dieser Zeit nicht für die Bearbeitung von Jobs des Clusters bereitstehen. Erst wenn die interaktive Arbeit pausiert oder beendet wurde und der Rechner sich im Leerlauf befindet, sollen Berechnungsjobs der Warteschlange ausgeführt werden.

Hierzu wird auf den ausführenden Rechnern der Keyboard-Dienst (*condor_kbdd*) verwendet, welcher die Maus und die Tastatur des jeweiligen Rechners überwacht. Wie im folgenden Quellcode dargestellt, dürfen Maus und Tastatur des Rechners für eine Dauer von 5 min (*StartIdleTime*) nicht genutzt werden und die CPU muss sich im Leerlauf befinden, damit die *START*-Anweisung ausgeführt wird und der Rechner die Jobs aus der Warteschlange bearbeitet.

```

1 # Konfigurationsdatei Execute-Rechners (condor_config)
2 # Dienste
3 DAEMON_LIST = MASTER SCHEDD STARTD KBDD
4
5 # Ruhezeit von Maus und Tastatur, danach START Job
6 StartIdleTime = 5 * $(MINUTE)
7 START = ( $(CPUIdle) && (KeyboardIdle > $(StartIdleTime)) )

```

Wird der Rechner interaktiv genutzt, erhält er den Status *Owner*, ist er bereit, Jobs zu bearbeiten, den Status *Unclaimed* (s. Abschnitt 5.3.3).

Weiterhin soll das interaktive Arbeiten an den Rechnern auch möglich sein, wenn auf ihnen bereits Jobs laufen, da von außen nicht erkennbar ist, welcher Rechner zur Zeit einen Job ausführt und die interaktive Arbeit nicht beeinträchtigt werden soll. Meldet sich ein Anwender an einem Rechner an, welcher einen Job der Warteschlange bearbeitet, und benutzt die Maus oder die Tastatur, wird der laufende Job noch maximal 5 min weiter bearbeitet (*BeforeSuspendTime*) und dann pausiert. Diese Zeit basiert auf Beobachtungen der Nutzeranmeldung in dem zur Umsetzung verwendeten Cluster. Neben der Anmeldung am Betriebssystem erfolgt in dieser Zeit das Starten der benötigten Software durch den Anwender. Es findet also noch keine wirkliche Arbeit an dem Rechner

statt und der Rechner kann den laufenden Job weiterhin ausführen. Durch diese weitere Bearbeitung des Jobs wird die Wahrscheinlichkeit der erfolgreichen Fertigstellung erhöht.

Wird die interaktive Arbeit für mindestens 5 min unterbrochen (`ContinueIdleTime`) oder beendet und befindet sich die CPU im Leerlauf, wird der Job fortgesetzt. Befindet sich der Job länger als 10 min im pausierten Zustand (`MaxSuspendTime`), wird er von dem Rechner entfernt und erneut der Warteschlange hinzugefügt. In diesem Fall geht der bisher erreichte Berechnungsfortschritt verloren. Die Umsetzung dieser Vorgehensweise ist im folgenden Quellcode dargestellt:

```

1 # Konfigurationsdatei des Execute-Rechners (condor_config)
2 # Pausieren von Jobs
3 WANT_SUSPEND = True
4 BeforeSuspendTime = 5 * $(MINUTE)
5 SUSPEND = ( $(KeyboardBusy) && ($(StateTimer) > $(BeforeSuspendTime)) )
6
7 # Fortsetzen von Jobs
8 ContinueIdleTime = 5 * $(MINUTE)
9 CONTINUE = ( $(CPUIidle) && ($(ActivityTimer) > 10) \
10             && (KeyboardIdle > $(ContinueIdleTime)) )
11
12 # Entfernen von Jobs
13 WANT_VACATE = False
14 MaxSuspendTime = 10 * $(MINUTE)
15 PREEMPT = ( ((Activity == "Suspended") && \
16             $(ActivityTimer) > $(MaxSuspendTime))) \
17             || (SUSPEND && (WANT_SUSPEND == False)) )

```

Durch die beschriebene Konfiguration der ausführenden Rechner können diese sehr flexibel auf externe Einflüsse wie die spontane interaktive Verwendung reagieren. Für den Einsatz des HTCCondor-Clusters bei der Lösung von Optimierungsaufgaben ist die reine Berücksichtigung der zur Verfügung stehenden Hardware nicht ausreichend. Zusätzlich müssen die auf den jeweiligen Rechnern installierte Software und die Anzahl der verfügbaren Lizenzen berücksichtigt werden. Erst wenn alle Voraussetzungen eines Jobs erfüllt sind, erfolgt dessen Bearbeitung.

Zur Beschreibung der auf einem Rechner installierten Software werden die Klassenattribute verwendet (s. Abschnitt 5.3.1). Für jedes relevante Programm auf einem Rechner wird eine neues Klassenattribut definiert, welches die jeweilige Softwarebezeichnung und den Boole'schen Wert *True* trägt. Neben der Erstellung müssen die Attribute zusätzlich den bereits vorhandenen Attributen hinzugefügt werden. Der folgende Quellcode stellt die benötigte Erweiterung der Konfigurationsdatei des Execute-Rechners (`condor_config`) unter der Verwendung des CAx-Systems NX (Version 10) und des Python-Interpreters (Version 3.4) dar:

```

1 # Konfigurationsdatei des Execute-Rechners (condor_config)
2 NX10 = True
3 Python34 = True
4 STARTD_ATTRS = $(STARTD_ATTRS), NX10, Python34

```

Weiterhin muss in der Jobdefinitionsdatei festgelegt werden, dass die definierten Attribute zur Bearbeitung des Jobs erfüllt sein müssen. Hierzu wird die vordefinierte Variable `requirements` verwendet. Die Verwendung der Variable ist im folgenden Quellcode dargestellt:

```

1 # Jobdefinitionsdatei
2 requirements = (NX10 == True) && (Python34 == True)

```

Der Job wird in diesem Fall nur ausgeführt, wenn diese Variable den Boole'schen Wert *True* annimmt. Dazu müssen beide Bedingungen (NX10 und Python34) erfüllt sein. Durch den Vergleichsoperator `==` wird die Unterscheidung von Groß- und Kleinschreibung sichergestellt. Soll diese Unterscheidung nicht gelten, muss der Vergleichsoperator `==` gewählt werden.

Mit Hilfe der Variable `requirements` können weitere Anforderungen eines Jobs an einen ausführenden Rechner gestellt werden. Die folgenden Klassenattribute sind bereits standardmäßig auf jedem Rechner definiert [Dzi15]:

- Rechnerarchitektur (`Arch: INTEL, X86_64, IA64, ...`)
- Betriebssystem (`OpSys: LINUX, OSX, WINDOWS, ...`)
- CPU-Anzahl (`Cpus`)
- Arbeitsspeicher (`Memory`)
- Rechenleistung (`KFlops`)

Zusätzlich können Mindestanforderungen definiert werden, was vor allem für die Spezifikation der benötigten Hardware sinnvoll ist, da hier in den meisten Fällen nicht genau ein Wert, sondern ein Mindestwert erfüllt sein muss, z. B. bei der Anzahl der CPU oder eines bestimmten Arbeitsspeichers. Die Mindestanforderungen eines Jobs werden in der Jobdefinitionsdatei durch den Befehl `request_<Anforderung>` festgelegt (z. B. `request_cpus = 8`). Die nachfolgend aufgelisteten Klassenattribute sind standardmäßig auf jedem Rechner definiert:

- Anzahl von CPUs (`request_cpus = <n>`)
- Arbeitsspeicher (`request_memory = <n>`)
- Festplattenkapazität (`request_disk = <n>`)
- Grafikprozessoren (`request_GPUs = <n>`)

Neben der auf den Rechnern des Clusters vorhandenen Software müssen auch die zur Verfügung stehenden Lizenzen als Ressourcen betrachtet werden. Bei der in der Produktentwicklung verwendeten Software haben sich auf Basis einer nutzungsunabhängigen Bemessungsgrundlage die folgenden Lizenzmodelle etabliert [LB09]:

- Named-User-Lizenzmodell bzw. Server/Rechner-Lizenzmodell: Bei diesem Lizenzmodell sind die Nutzungsrechte der Software an einen spezifischen Anwender bzw. einen spezifischen Server oder Rechner gebunden. Dieses Lizenzmodell wird z. B. bei Microsoft Office Produkten verwendet [Häu11]. Meistens wird die Lizenz bei der Installation einmalig konfiguriert und es wird nur eine bestimmte Anzahl von Installationen der Software zugelassen. Ist also eine Software auf einem Rechner installiert, verfügt dieser auch über eine gültige Lizenz. Dieses Lizenzmodell wird durch die bereits beschriebenen Softwareanforderungen eines Jobs abgedeckt.

- Concurrent-User-Lizenzmodell: Das Concurrent-User-Lizenzmodell ist nicht anwender- oder rechnerbezogen, sondern es wird die Anzahl der Anwender limitiert, welche die Software zu einem Zeitpunkt gleichzeitig nutzen dürfen. Somit kann eine Software auf mehreren Rechnern installiert sein, aber nur auf einer begrenzten Anzahl von Rechnern gleichzeitig genutzt werden. Dieses Lizenzmodell wird z. B. bei der ERP-Software SAP sowie den gängigen CAD- und CAE-Systemen verwendet [Häu11]. Da die einzelnen Lizenzen bei diesem Modell fließend an die jeweiligen Rechner vergeben werden, werden die Lizenzen auch als Floating-Lizenzen bezeichnet.

Zur Realisierung des Concurrent-User-Lizenzmodells wird in dem Cluster die Anzahl der gleichzeitig laufenden Jobs begrenzt, welche eine bestimmte Software benötigen. Hierzu wird die Konfigurationsdatei des verwaltenden Rechners (Central Manager) um den Befehl `LIMIT` erweitert [Dzi15]. Der folgende Quellcode zeigt die Erweiterung der Konfigurationsdatei wenn für das CAx-System NX (Version 10) nur fünf Floating-Lizenzen zur Verfügung stehen:

```
1 # Konfigurationsdatei des Central Managers (condor_config)
2 NX10_LIMIT = 5
```

In der Jobdefinitionsdatei werden ebenfalls die zu berücksichtigenden Limits definiert. Sollen mehrere Floating-Lizenzen berücksichtigt werden, müssen die Bezeichnungen durch ein Komma abgetrennt werden. Der folgende Quellcode zeigt die Erweiterung der Jobdefinitionsdatei für das CAx-System NX.

```
1 # Jobdefinitionsdatei
2 concurrency_limits = NX10
```

Durch die beschriebenen Befehle innerhalb der jeweiligen Konfigurations- und Jobdefinitionsdateien werden in dem Cluster sowohl die zur Verfügung stehende Hardware als auch die vorhandene Software und die verfügbaren Lizenzen bei der Bearbeitung von Jobs berücksichtigt. Der Cluster bildet somit die wesentlichen Anforderungen zur parallelen Bearbeitung von Optimierungsaufgaben in der Produktentwicklung ab.

5.3.7 Priorisierung von Ressourcen

Neben der Erfüllung der Anforderungen zum Ausführen eines Jobs ist es sinnvoll, die Reihenfolge der zu verwendenden Ressourcen festzulegen, um somit leistungsstärkere Rechner zu bevorzugen und die Laufzeit der Jobs möglichst gering zu halten. Dazu stehen zwei Möglichkeiten zur Verfügung: die Festlegung einer generellen Reihenfolge auf der Basis von Klassenattributen und die Einteilung der Rechner in Vorzugsgruppen [Dzi15].

Eine generelle Reihenfolge zur Verwendung der zur Verfügung stehenden Rechner wird durch den Befehl `rank` festgelegt. Dadurch wird vorab eine quantitative Bewertung der Rechner vorgenommen. Diese Bewertung kann anhand der folgenden Größen erfolgen [Dzi15]:

- CPU-Anzahl (`Cpus`)
- Arbeitsspeicher (`Memory`)
- Rechenleistung (`KFlops`)

Durch den Befehl `rank` wird in der Jobdefinitionsdatei die Reihenfolge festgelegt, nach der die zur Verfügung stehenden Rechner ausgewählt werden. Insbesondere für die Durchführung von FEM-Berechnung ist ein möglichst großer Arbeitsspeicher vorzuziehen. Wird der folgende Quellcode in der Jobdefinitionsdatei verwendet, wird immer der Rechner mit dem größten Arbeitsspeicher bevorzugt.

```
1 # Jobdefinitionsdatei
2 rank = memory
```

Damit die festgelegte Reihenfolge auch bei der Zuweisung der Ressourcen berücksichtigt wird, muss in der Konfiguration des Negotiator-Dienstes des verwaltenden Rechners (s. Abschnitt 5.3.1) die folgende Anweisung verwendet werden:

```
1 # Konfigurationsdatei des Central Managers (condor_config)
2 NEGOTIATOR_PRE_JOB_RANK = MY.RANK
```

Dadurch werden die Rechner, welche die Anforderungen zum Ausführen eines Jobs erfüllen, zunächst anhand des in der Jobdefinitionsdatei festgelegten Ranges sortiert und der Rechner mit dem höchsten Wert ausgewählt.

Die zweite Möglichkeit zur Priorisierung der Ressourcen ist die Einteilung der Rechner in Vorzugsgruppen. Diese Einteilung kann z. B. auf Basis der Leistung der Rechner, des Standortes oder des Nutzungsverhaltens erfolgen. In den Konfigurationsdateien der jeweiligen ausführenden Rechner wird die Gruppenzugehörigkeit dieser Rechner definiert. Mittels des folgenden Quellcodes wird ein Rechner der Gruppe `Dell-T1600` hinzugefügt.

```
1 # Konfigurationsdatei des Execute-Rechners (condor_config)
2 MachineOwner = "Dell-T1600"
3 STARTD_ATTRS = $(STARTD_ATTRS), MachineOwner
4 RANK = TARGET.Group == MY.MachineOwner
```

Sollen die Rechner dieser Gruppe bei der Ausführung eines Jobs bevorzugt werden, muss dies in der Jobdefinitionsdatei festgelegt werden. Die Rechner der Gruppe werden dann durch den Befehl `RANK` höher bewertet als Rechner anderer Gruppen oder Rechner ohne Gruppenzugehörigkeit [Dzi15]. Soll die Gruppe `Dell-T1600` bei der Durchführung eines Jobs bevorzugt werden, muss in der Jobdefinitionsdatei die folgende Anweisung verwendet werden:

```
1 # Jobdefinitionsdatei
2 +Group = "Dell-T1600"
```

Die beiden Methoden zur Priorisierung der vorhandenen Ressourcen können unabhängig voneinander genutzt werden. Werden beide Methoden in einer Jobdefinitionsdatei angegeben, hat die Gruppenzugehörigkeit stets Vorrang vor der generellen Reihenfolge auf Basis der Klassenattribute. Innerhalb der Vorzugsgruppe werden die Rechner anhand der Klassenattribute priorisiert.

5.3.8 Energieverwaltung des Clusters

Der HTCondor-Cluster soll allen Anwendern zur Durchführung von Optimierungen und anderen Berechnungen durchgehend zur Verfügung stehen. Nicht verwendete Rechner

sollen hierbei nicht permanent im Betrieb sein, sondern selbstständig einen Zustand niedrigeren Energieverbrauchs einnehmen und bei Bedarf wieder aus diesem in den normalen Betriebszustand wechseln.

Dazu müssen die Klassenattribute der Rechner im Energiesparmodus dem verwaltenden Rechner bekannt sein, damit diese bei der Jobvergabe mit den Anforderungen der Jobs abgeglichen werden können. Diese Informationen werden in einer Log-Datei (`OfflineLog`) gespeichert, welche vom verwaltenden Rechner abgerufen werden kann. Damit die Informationen nicht beliebig lange in dieser Datei gespeichert werden, werden sie nach einer Woche gelöscht (`OFFLINE_EXPIRE_ADS_AFTER = 3600*24*7`). Da für diesen Wert standardmäßig in HTCondor der größte positive Wert für einen 32-Bit vorzeichenbehafteten Integer in Sekunden verwendet wird (2.147.483.647), was einer Dauer von ca. 68 Jahren entspricht, muss der Wert reduziert werden, um zu verhindern, dass Informationen von evtl. nicht mehr vorhandenen Rechnern gespeichert werden. Zudem werden die Rechner, welche in den Energiesparmodus gewechselt sind, als *offline* markiert.

Das automatische Starten der Rechner, die sich im Energiesparmodus befinden, erfolgt durch den Rooster-Dienst (`condor_rooster`), welcher der Liste der Dienste hinzugefügt wird. Durch den Dienst wird im Intervall von 5 min (`ROOSTER_INTERVAL`) überprüft, ob Rechner im Energiesparmodus die Jobanforderungen erfüllen. Werden die Jobanforderungen durch einen Rechner erfüllt, erfolgt das Starten des Rechners durch das Programm `condor_power`. Weiterhin kann über den Befehl `ROOSTER_MAX_UNHIBERNATE` die Anzahl der Rechner definiert werden, welche pro Intervall gestartet werden. Der Standardwert dieses Befehls beträgt 0, was keiner Limitierung entspricht. Die beschriebenen Einstellungen in der Konfigurationsdatei des verwaltenden Rechners sind im folgenden Quellcode dargestellt:

```
1 # Konfigurationsdatei des Central Managers (condor_config)
2 # Logs fuer den Energiesparplan
3 OFFLINE_LOG = $(SPOOL)\OfflineLog
4 VALID_SPOOL_FILES = $(VALID_SPOOL_FILES), $(OFFLINE_LOG)
5 OFFLINE_EXPIRE_ADS_AFTER = 3600*24*7
6
7 # Dienst und Update-Intervall
8 DAEMON_LIST = MASTER SCHEDD COLLECTOR NEGOTIATOR ROOSTER
9 ROOSTER_INTERVAL = 5 * $(MINUTE)
10
11 # Wake On LAN
12 ROOSTER_UNHIBERNATE = Offline && Unhibernate
13 #ROOSTER_MAX_UNHIBERNATE = 1
14 ROOSTER_WAKEUP_CMD = "$(BIN)\condor_power.exe -d -i"
```

Neben dem verwaltenden Rechner müssen auch die ausführenden Rechner des Clusters konfiguriert werden. Befindet sich ein Rechner im Leerlauf und wird nicht interaktiv verwendet, wird dieser nach 60 min in den Ruhezustand (S4) versetzt. Die beschriebenen Bedingungen werden im Intervall von 5 min überprüft. Der folgende Quellcode zeigt die dazu notwendigen Anpassungen in der Konfigurationsdatei der ausführenden Rechner:

```

1 # Konfigurationsdatei des Execute-Rechners (condor_config)
2 # Zeitintervalle
3 HIBERNATE_CHECK_INTERVAL = 5 * $(MINUTE)
4 TimeToWait = 60 * $(MINUTE)
5
6 # Ruhezustand
7 ShouldHibernate = ( (KeyboardIdle > $(StartIdleTime)) \
8                     && $(CPUIdle) && (State == "Unclaimed" && Activity == "Idle") \
9                     && ($(StateTimer) > $(TimeToWait)) )
10 HibernateState = "S4"
11 HIBERNATE = ifThenElse($(ShouldHibernate), $(HibernateState), "NONE")

```

Ein Rechner kann unter dem Betriebssystem Windows verschiedene Zustände niedrigeren Energieverbrauchs einnehmen, welche sich primär in der Stromversorgung einzelner Komponenten des Rechners unterscheiden. Eine Übersicht der in Windows möglichen Energiezustände ist in [Cen15], [Mic16] zu finden. Der durch HTCondor herbeigeführte Ruhezustand des Rechners wird über die Variable `HibernateState` festgelegt.

Für die prototypische Umsetzung wird der Zustand S4 verwendet, da in diesem Zustand die meiste Energie eingespart wird und das externe Starten des Rechners durch ein Wake-On-LAN-Signal unterstützt wird. Im Gegensatz zum Zustand S3 benötigt der Rechner wenige Sekunden länger zum Hochfahren des Betriebssystems [Dzi15]. Da die ausführenden Rechner innerhalb einer Optimierung maximal einmal zu Beginn der Optimierung gestartet werden müssen, kann diese Zeit vernachlässigt werden. Der Zustand S5 kann für das externe Starten der ausführenden Rechner nicht verwendet werden, da in diesem Zustand keine Komponente des Rechners mit Strom versorgt wird und somit der Netzwerkadapter nicht zum Starten des Rechners angesteuert werden kann.

Bei der Auswahl der zu startenden Rechner wird zudem die festgelegte Priorisierung der Rechner berücksichtigt (s. Abschnitt 5.3.7). Befindet sich ein höher priorisierter Rechner im Ruhezustand, wird dieser gestartet und einem niedriger priorisierten Rechner im normalen Betriebszustand vorgezogen. Auch hierbei kann die Zeit zum Hochfahren des Betriebssystems aus den beschriebenen Gründen vernachlässigt werden.

5.3.9 Konfiguration externer Software

Kommunikation und Datenaustausch über das Netzwerk erfolgen in HTCondor über TCP/IP-Sockets. Dies ermöglicht die Kommunikation zwischen Prozessen auf mehreren Rechnern, ohne dass ein gemeinsames Datei- oder Betriebssystem vorhanden ist [EBS15]. Unter dem Betriebssystem Windows wird diese Schnittstelle durch die Winsock-API realisiert. Über diese Programmierschnittstelle kann ein sog. Layered Service Provider (LSP) in den TCP/IP-Socket implementiert werden. LSPs werden meistens als dynamische Programmbibliotheken (DLL) realisiert. Dabei kann es zu Konflikten zwischen den DLLs verschiedener Programme kommen, was zu Einschränkungen in der Funktionalität der Programme führen kann.

Durch die Verwendung sog. Web-Filter, welche bestimmte Inhalte des Internets (z. B. Webseiten) blockieren, kann es zu Einschränkungen in der Funktionalität und zu Stabilitätsproblemen von HTCondor kommen. So werden z. B. einzelne Dienste nicht gestartet bzw. beendet und Rechner verlieren die Verbindung zum Cluster. Beim Auftreten derartiger Fehler sollten die LSPs eines Rechners überprüft werden. Dazu kann z. B. das Programm LSP-Fix genutzt werden, welches die LSPs eines Rechners untersucht und erkannte

Probleme auflistet. Weiterhin ermöglicht das Programm das Entfernen der Problem verursachenden DLLs [Cou16]. Die Benutzungsoberfläche des Programms ist in Abbildung 5.8 dargestellt.

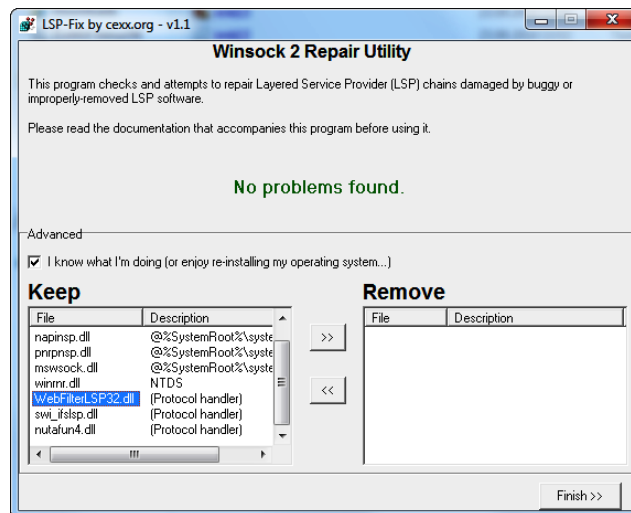


Abbildung 5.8: LSP-Fix zur Überprüfung der LSPs eines Rechners

In dem zur prototypischen Umsetzung genutzten Cluster führen zwei LSPs zu Stabilitätsproblemen von HTCondor: *WebFilterLSP32.dll* und *swi_ifslsp.dll*. Der erste LSP realisiert den Web-Filter der Klassenraum-Management-Software Netop Vision, der zweite den Web-Filter der Virenschutz-Software Sophos. Beide Web-Filter können in den jeweiligen Programmen deaktiviert und die LSPs entfernt werden. Es entsteht kein Sicherheitsrisiko, da lediglich die Blockierung bestimmter Webseiten aufgehoben wird. Durch das Entfernen der genannten LSPs kann ein stabil und robust arbeitender Cluster gewährleistet werden.

5.4 Optimierungssystem DAG2OPT

Die Schnittstelle des entwickelten Frameworks zum Anwender bildet das Optimierungssystem DAG2OPT. Bis auf das Erstellen der Skripte zur Automatisierung des Evaluationsprozesses können alle Aktivitäten zur Durchführung einer Optimierung über eine grafische Benutzungsoberfläche durchgeführt werden. DAG2OPT stellt somit das Front-End des gesamten Frameworks dar und realisiert die Generierung des Evaluationskriptes und die Kommunikation zwischen den einzelnen Modulen. Da hierbei die Einbindung verschiedener Evaluationsprozesse möglich ist sowie das Evaluations- und das Optimierungsmodell unabhängig voneinander agieren können, wird diese Form des Optimierungssystems auch als externes Optimierungssystem bzw. *General Purpose*-Optimierer bezeichnet [Har14].

Die auf Basis einer Recherche frei zugänglicher grafischer Modellierungssoftware (s. Tabelle A.9 im Anhang A.5) ausgewählte Open-Source-Software *Depends* [GU14] bildet die Grundlage für die Entwicklung der Benutzungsoberfläche von DAG2OPT, welche in Abbildung 5.9 dargestellt ist.

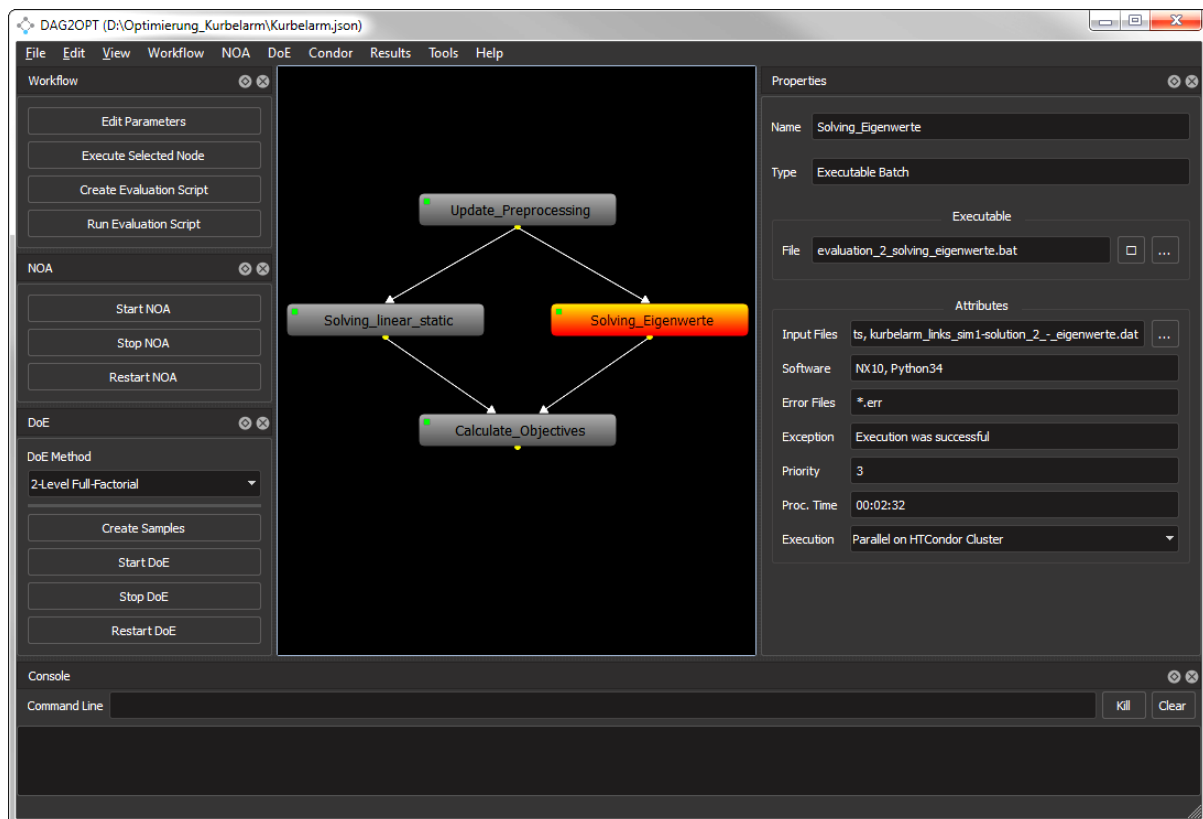


Abbildung 5.9: Benutzungsoberfläche des Optimierungssystems DAG2OPT

Ein Grafikbereich in der Mitte der Benutzungsoberfläche dient zur Definition des Evaluationsprozesses als DAG. Alle weiteren Fenster können flexibel um diesen zentralen Grafikbereich angeordnet oder auch ausgeblendet werden. Steuernde und ausführende Werkzeuge sind in der Abbildung im linken Bereich der Benutzungsoberfläche angeordnet. Im Bereich *Workflow* können neben der Editierung von Parametern einzelne Prozesselemente des DAG ausgeführt, das Evaluationskript erstellt oder der gesamte Evaluationsprozess gestartet werden, z. B. für Testzwecke. In den weiteren Bereichen wird eine Optimierung unter Verwendung des Optimierungssystems NOA [JC04] oder eine DoE-Studie gesteuert. Die Definition benötigter Informationen der Prozesselemente des Evaluierungsmodells erfolgt über den rechts angeordneten Bereich (*Properties*). Zur Überwachung einer laufenden Optimierung bzw. DoE-Studie dient eine Kommandokonsole im unteren Bereich. Hier werden sämtliche Ausgaben der bei der Evaluation verwendeten Software angezeigt. Darüber hinaus besteht die Möglichkeit, eigene Befehle in der Konsole auszuführen. Weitere Funktionen wie z. B. Einstellungen und das Monitoring des HTCCondor-Clusters (s. Abschnitt 5.3.3) können über die Menüleiste aufgerufen werden.

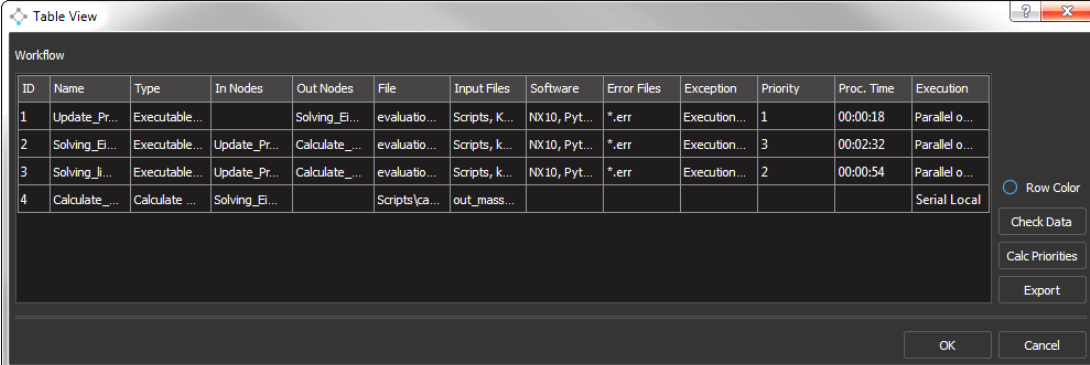
In den folgenden Abschnitten werden die Definition des Optimierungs- und Evaluationsmodells sowie die Optimierung der Prozessprioritäten beschrieben. Auf Basis von drei Testszenarien unterschiedlicher Komplexität werden verschiedene Scheduling-Verfahren gemäß ihrer Eignung zur Ermittlung der optimalen Prozessprioritäten analysiert und anhand der Effektivität ein Scheduling-Verfahren ausgewählt, welches in das Optimierungssystem implementiert wird.

5.4.1 Definition des Optimierungs- und Evaluationsmodells

Bei der Definition von Evaluationsprozessen wird allgemein zwischen Prozessströmen und Datenströmen unterschieden. Der Prozessstrom beschreibt die Abhängigkeiten der einzelnen Prozesselemente untereinander, der Datenstrom die Verarbeitung der relevanten Daten [LM12]. Da gemäß des Concurrent und Simultaneous Engineering ausschließlich die Verfügbarkeit und die Stabilität der benötigten Informationen für die Verarbeitung und die Parallelisierung von Prozessen relevant sind, ergibt sich die Abhängigkeit der einzelnen Evaluationsprozesse aus dem Datenstrom, welcher somit den Prozessstrom generiert. Eine explizite Trennung und Definition dieser beiden Ströme ist somit nicht notwendig.

Die Definition des Evaluationsprozesses durch den Anwender ist aus Prozesssicht deutlich übersichtlicher und nachvollziehbarer als eine reine Definition der relevanten Daten. Daher wird für die Prozessdefinition eine grafische Repräsentation durch einen DAG gewählt, welcher im zentralen Grafikbereich erstellt wird. Neue Prozesselemente werden über die rechte Maustaste hinzugefügt und die Abhängigkeiten der Elemente gemäß der DAG-Beschreibung durch Pfeile definiert. Die zur Bearbeitung notwendigen Metainformationen werden über die Eigenschaften im rechten Bereich der Benutzungsoberfläche für jedes Prozesselement festgelegt (s. Abbildung 5.9). Zu den Metainformationen zählen die Zuweisung einer auszuführenden Datei oder eines Skriptes, die zur Verarbeitung benötigten Eingabedateien und Software sowie die Festlegung fehlerbeschreibender Dateien und Ausnahmen. Weiterhin können die Priorität des Prozesselementes und die Laufzeit manuell eingegeben und editiert werden. Diese Werte werden in der Regel durch das System ermittelt. Bei der Auswahl der Bearbeitungsart (*Execution*) stehen zwei Möglichkeiten zur Verfügung: die lokale Bearbeitung auf dem aktuellen Rechner oder die Bearbeitung in dem HTCondor-Cluster. Somit kann auch solche Software für die Evaluation genutzt werden, welche für die Ausführung eine grafische Ausgabe benötigt und keinen Batch-Modus beinhaltet. Basierend auf den erstellten Metainformationen wird das Evaluationskript für die Optimierung erstellt. Dieses Skript enthält die vollständige Logik zur Steuerung der Evaluationsprozesse und zur Bereitstellung der flexiblen Parallelisierung der Prozesselemente.

Alternativ zur rein grafischen Repräsentation kann der Evaluationsprozess zusätzlich in einer Tabelle angezeigt und editiert werden. Auch die Optimierung der Prozessprioritäten (s. Abschnitt 5.4.2) kann aus dieser Ansicht heraus aufgerufen werden. Die Tabellenansicht eines Evaluationsprozesses ist in Abbildung 5.10 dargestellt.



ID	Name	Type	In Nodes	Out Nodes	File	Input Files	Software	Error Files	Exception	Priority	Proc. Time	Execution
1	Update_Pr...	Executable...		Solving_EI...	evaluatio...	Scripts, K...	NX10, Pyt...	*.err	Execution...	1	00:00:18	Parallel o...
2	Solving_EI...	Executable...	Update_Pr...	Calculate_...	evaluatio...	Scripts, k...	NX10, Pyt...	*.err	Execution...	3	00:02:32	Parallel o...
3	Solving_IJ...	Executable...	Update_Pr...	Calculate_...	evaluatio...	Scripts, k...	NX10, Pyt...	*.err	Execution...	2	00:00:54	Parallel o...
4	Calculate_...	Calculate ...	Solving_EI...		Scripts\ca...	out_mass...						Serial Local

Abbildung 5.10: Tabellendarstellung der Evaluationsprozesse in DAG2OPT

Die Schnittstelle zwischen dem Optimierungs- und dem Evaluationsmodell bilden die Designvariablen, welche in DAG2OPT über eine Tabelle in einem separaten Dialog definiert werden. Dazu werden der Name, die Ober- und Untergrenze sowie die Schrittweite bzw. die Schrittzahl einer Designvariable festgelegt. Zudem ist die Eingabe konkreter Werte möglich, um somit die Evaluation einer bestimmten Produktvariante durchzuführen. Darüber hinaus stehen dem Anwender verschiedene Funktionen für den Import und Export von Designvariablen zur Verfügung. Der Dialog zur Definition der Designvariablen ist in Abbildung 5.11 dargestellt.

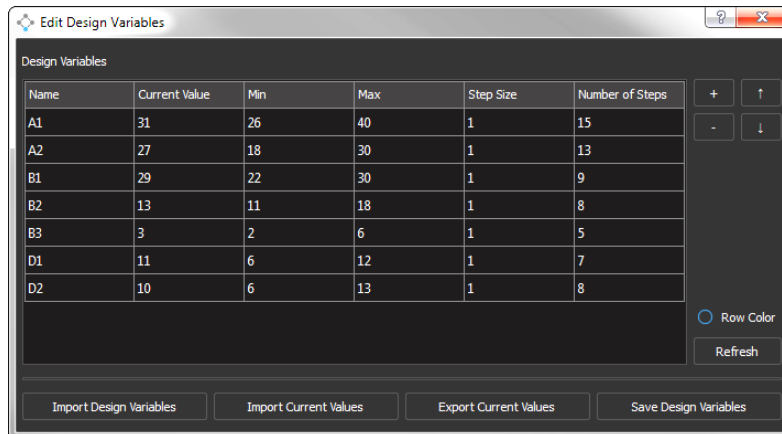


Abbildung 5.11: Definition der Designvariablen in DAG2OPT

Bei einer laufenden Optimierung oder der Evaluation einer Produktvariante werden die aktuellen Werte der Designvariablen in eine Datei (parameter.txt) geschrieben. Diese Datei muss bei der Definition des Evaluationsmodells durch den Anwender als Input-Datei des ersten Prozesselementes des DAG ausgewählt werden. Im ersten Prozesselement werden somit in der Regel die Parameter eingelesen und ggf. auch direkt daraus ein Modell der zu evaluierenden Produktvariante erstellt.

Die zweite Schnittstelle zwischen dem Evaluations- und dem Optimierungsmodell bildet die Berechnung des Zielfunktionswertes aus der Zielfunktion. Diese Berechnung wird als letztes Prozesselement (*Calculate Objectives*) des DAG definiert. Hier müssen alle vorigen Prozesselemente zusammenlaufen und deren Ergebnisdateien als Input-Dateien festgelegt werden. Die Definition der Zielfunktion erfolgt in einem separaten Dialog, welcher in Abbildung 5.12 dargestellt ist.

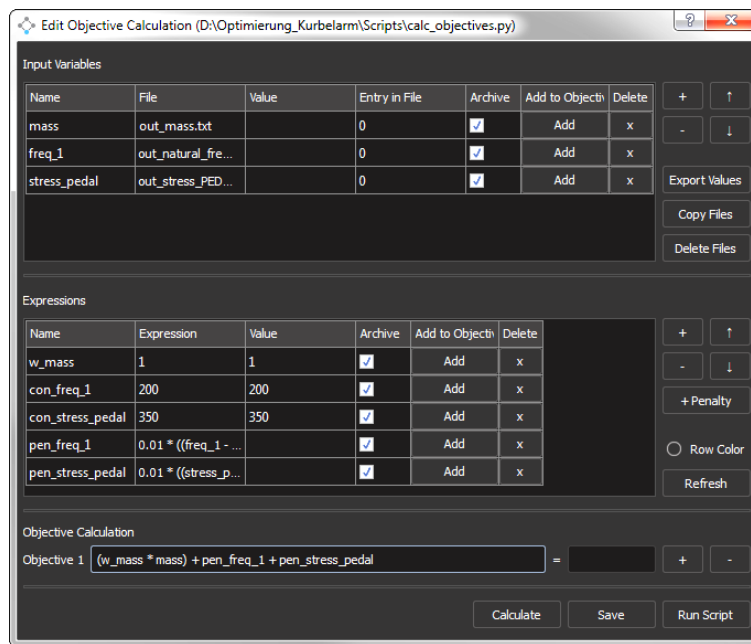


Abbildung 5.12: Definition der Zielfunktion in DAG2OPT

Der Dialog ist in drei Bereiche unterteilt: die Definition der Input-Variablen, die Erstellung zusätzlicher Ausdrücke und die Definition der Zielfunktion. Input-Variablen repräsentieren die Ergebnisgrößen aus den Input-Dateien der Zielfunktionsberechnung. Um den Programmieraufwand zum Einlesen einer Input-Variable zu reduzieren, wird anhand der Eingabe eines numerischen Wertes über einen internen Parser der Wert der Input-Variable aus der Datei extrahiert. Ein Parser ist ein Programm oder eine Programmroutine, mit der eine Dateneingabe zerlegt und die Daten für die Weiterverarbeitung in ein geeignetes Format umgewandelt werden [FH11]. Existieren in einer Input-Datei mehrere Einträge, wird der Wert anhand seiner Position in der Datei extrahiert und eingelesen.

Die Input-Variablen können nach dem Einlesen in Ausdrücken (*Expressions*) verwendet werden, z. B. bei der Verwendung von Straffunktionen (s. Abschnitt 2.8.1). Auch das Erstellen zusätzlicher Ausdrücke, z. B. Gewichtungsfaktoren, ist an dieser Stelle möglich. Die Input-Variablen und Ausdrücke stehen anschließend zur Verwendung bei der Zielfunktionsformulierung im unteren Bereich zur Verfügung. Bei Verwendung eines multi-kriteriellen Optimierungsalgorithmus besteht die Möglichkeit, mehrere Zielfunktionen zu erstellen. Weiterhin kann für jede Input-Variable und jeden Ausdruck festgelegt werden, ob deren Wert bei einer Optimierung oder DoE-Studie archiviert werden soll.

Aus den vorgenommenen Einträgen wird beim Speichern oder bei der Beendigung des Dialogs ein Python-Skript [Py16] generiert, durch welches der Zielfunktionswert berechnet und in eine Datei geschrieben wird (objectives.txt). Für Testzwecke kann dieses Skript auch direkt aus dem Dialog heraus aufgerufen werden.

Das Evaluationsmodell kann somit vollständig innerhalb der Benutzungsoberfläche von DAG2OPT erstellt werden. Zum Editieren von Skripten der einzelnen Prozesselemente steht in dem System ein Texteditor inklusive Syntax-Hervorhebung zur Verfügung. Die Schnittstellen zwischen dem Optimierungs- und dem Evaluationsmodell bilden die Parameterdatei (parameter.txt) und die Zielfunktionswertdatei (objectives.txt).

5.4.2 Optimierung der Prozessprioritäten

Durch die prioritätenbasierte Bearbeitung in Verbindung mit dynamischer Allokation wird bei der parallelen Bearbeitung von Evaluationsprozessen die Laufzeit verkürzt und somit die Effizienz der Optimierung gesteigert. Dabei stellt vor allem die Ermittlung der optimalen Prioritäten der Prozesselemente eine Herausforderung dar, für deren Lösung verschiedene Scheduling-Verfahren zur Verfügung stehen (s. Abschnitt 4.5).

Anhand von drei Testszenarien unterschiedlicher Komplexität werden in diesem Abschnitt verschiedene Scheduling-Verfahren analysiert und gemäß ihrer Ergebnisqualität bewertet. Dabei werden zwei Testszenarien für die Untersuchung des Verhaltens und die Entwicklung eines Werkzeugs zur Ermittlung der optimalen Prozessprioritäten verwendet und die Funktionsweise des Werkzeugs an einem dritten Testszenario überprüft. Mittels der vollständigen Enumeration werden vorab alle möglichen Kombinationen von Prioritäten untersucht und somit der gesamte Lösungsraum ermittelt. Abbildung 5.13 zeigt die Testszenarien, welche für die Untersuchung des Verfahrens zur Optimierung der Prozessprioritäten verwendet werden.

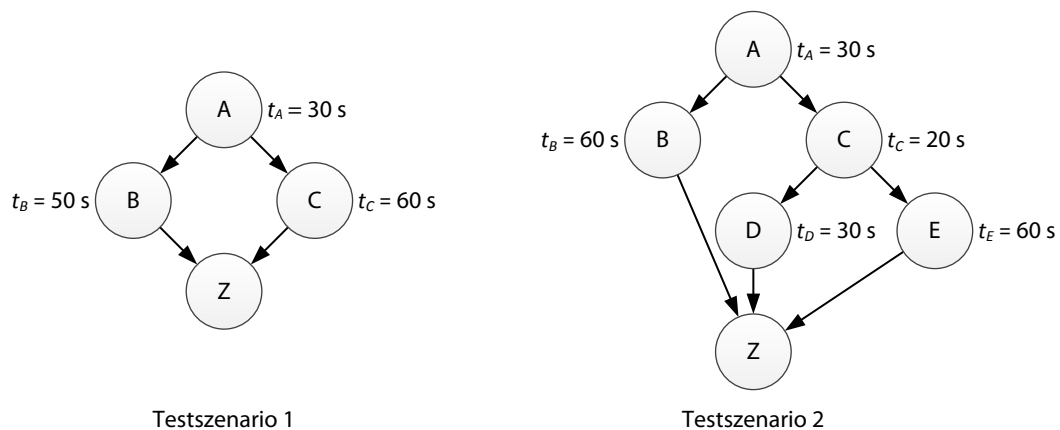


Abbildung 5.13: Testszenarien für die Bestimmung des Verfahrens zur Optimierung der Prozessprioritäten

Testszenario 1 stellt einen DAG bestehend aus drei Prozesselementen (A, B, C) zur Evaluation dar, welche in der Berechnung des Zielfunktionswertes (Z) zusammenlaufen. Dieser DAG kann repräsentativ für den in Abbildung 5.9 dargestellten Evaluationsprozess gesehen werden. Neben den Prozesselementen ist die Laufzeit des jeweiligen Elementes dargestellt. Die Berechnung des Zielfunktionswertes aus den Zustandsvariablen erfolgt sehr schnell und muss daher nicht bei der Ermittlung der Prioritäten berücksichtigt werden. Der DAG in Testszenario 2 enthält eine zusätzliche Abhängigkeitsstufe und bildet einen komplexeren Anwendungsfall. Auch hier laufen die jeweils letzten Prozesselemente eines Pfades (B, D, E) in der Berechnung des Zielfunktionswertes (Z) zusammen.

Um die Verfahren zur Ermittlung der optimalen Prozessprioritäten zu untersuchen, wird zunächst das Allokationsverhalten von HTCCondor analysiert und ein Modell zur Simulation der Allokation erstellt. Hierzu wurden in [Cöl16] diverse Versuche zur Allokation von HTCCondor durchgeführt und fünf Regeln abgeleitet, welche das Allokationsverhalten beschreiben (s. Anhang A.6.2). Das aus diesen Regeln erstellte Simulationsmodell bildet die idealisierte Funktionsweise von HTCCondor ab. Zufällig auftretende Störeinflüsse durch

externe Prozesse der einzelnen Slots werden dabei nicht berücksichtigt. Das Aktivitätsdiagramm der Allokationssimulation von HTCCondor ist in Abbildung A.5 im Anhang A.6.3 dargestellt.

Bevor die Simulation der Allokation durchgeführt wird, werden die benötigten Daten importiert. Hierzu zählen die Daten des zur Verfügung stehenden Clusters, welche aus HTCCondor ausgelesen werden und neben den Slots bzw. Rechnern auch die vorhandene Software und die Lizenzen beinhalten. Weiterhin werden die DAG-Informationen inklusive der Metadaten der Prozesselemente (s. Abschnitt 5.4.1) sowie die Größe der Parallelisierungsdomäne n_p (z. B. Populationsgröße eines GA, $n_p = n_I$) benötigt. Diese werden direkt durch DAG2OPT selbst bereitgestellt. Das Ergebnis der Allokationssimulation ist die maximale Laufzeit t zur Durchführung der Evaluationsprozesse auf den zur Verfügung stehenden Ressourcen. Diese Laufzeit soll durch die Ermittlung der optimalen Prozessprioritäten minimiert werden.

Die Ergebnisse der Allokationssimulation im Vergleich mit der Laufzeit der realen Allokation von HTCCondor sind für das Testszenario 1 in Abbildung 5.14 dargestellt. Die Anzahl der verwendeten Slots bzw. Rechner n_s beträgt 4.

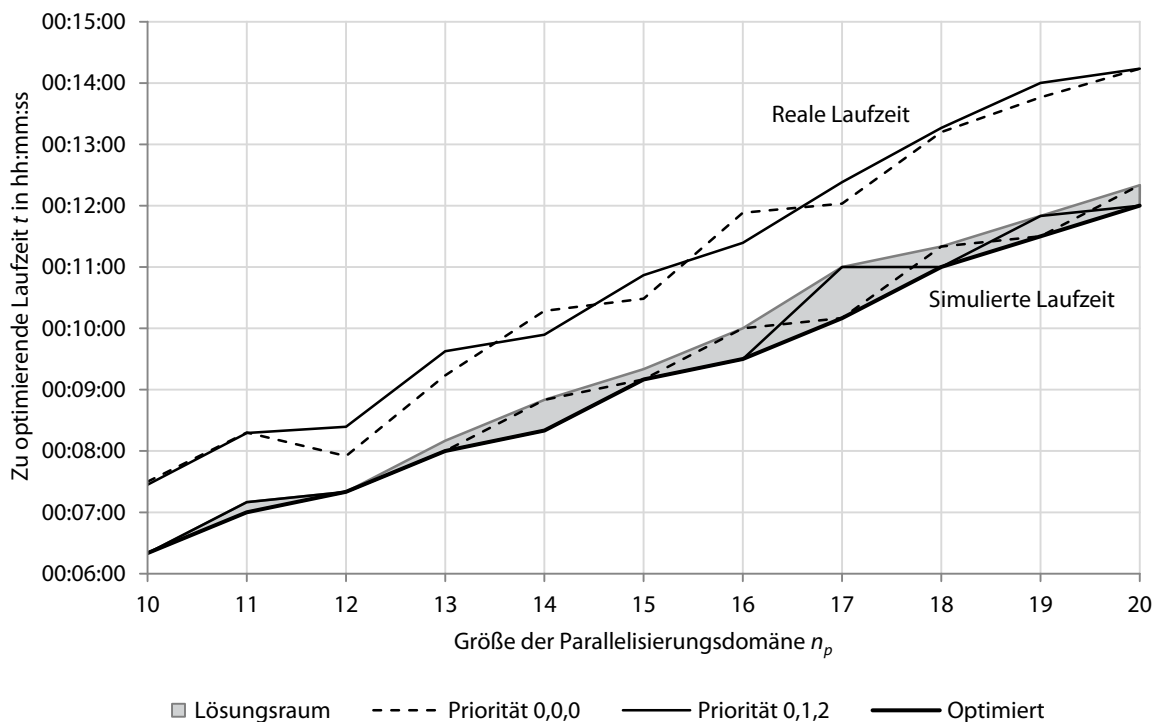


Abbildung 5.14: Vergleich von simulierter und realer Laufzeit für Testszenario 1 ($n_s = 4$)

Die realen Laufzeiten basieren jeweils auf dem Mittelwert aus fünf Durchläufen aufgrund der Streuung der Messwerte durch unregelmäßig auftretende externe Prozesse der einzelnen Slots. Die Allokation von HTCCondor wird qualitativ sehr gut durch das Simulationsmodell abgebildet. Beide Laufzeiten steigen mit der Größe der Parallelisierungsdomäne n_p linear an und die grundlegende Charakteristik der Allokation wird durch das Simulationsmodell wiedergegeben. Die exakte quantitative Wiedergabe der realen Laufzeiten durch die Allokationssimulation ist aufgrund der Streuung der realen Messwerte sehr schwierig. Obwohl bei der Allokationssimulation eine mittlere gemessene Overhead-Zeit von 10s be-

rücksichtigt wird, (s. Anhang A.6.1), beträgt die mittlere Differenz zwischen realer und simulierter Laufzeit 18 %.

Weiterhin wird der Einfluss der Prozessprioritäten untersucht. Dabei werden in der ersten Variante ausschließlich Prioritätswerte von 0 verwendet. Die Prozesselemente werden somit rein nach dem FIFO-Prinzip bearbeitet. Die zweite Variante bildet die Heuristik des *kritischen Pfades* bzw. *Longest Path Following* ab (s. Abschnitt 4.5). Hierbei werden die Laufzeiten der Elemente eines Pfades addiert und der Pfad mit der längsten Laufzeit am höchsten priorisiert. Für das Testszenario 1 ist die daraus resultierende Priorisierung identisch mit der Scheduling-Heuristik *Longest Processing Time*. Die Prozesselemente werden gemäß der alphabetischen Reihenfolge der Bezeichnung in Abbildung 5.13 mit den Werten 0, 1, 2 priorisiert.

Da bei der Verwendung einer Heuristik nicht eindeutig festgestellt werden kann, ob die mit der Heuristik ermittelten Prioritäten auch zur minimalen Laufzeit führen, wird auf Basis des Simulationsmodells der komplette Lösungsraum des Problems durch vollständige Enumeration ermittelt. Hierbei werden die Laufzeiten aller möglichen Kombinationen der Prioritäten ermittelt. Im Gegensatz zur Simulation, welche nur wenige Sekunden benötigt, wird bei der realen Allokation die gesamte Laufzeit für die Allokation der Prozesselemente benötigt. Da die reale Untersuchung aller Kombinationen der Prioritätswerte sehr zeitaufwendig ist und die Laufzeiten zudem Streuungen unterliegen, wird auf die Ermittlung des realen Lösungsraumes verzichtet.

Bei der Variation der Größe der Parallelisierungsdomäne n_p und der Betrachtung des Lösungsraumes ist zu erkennen, dass mit zunehmender Größe der Parallelisierungsdomäne der Einfluss der Prioritäten auf die Laufzeit zunächst ansteigt und dann nahezu konstant verläuft. Der maximale Einfluss zeigt sich hier bei $n_p = 17$. Die Differenz zwischen der längsten und der kürzesten Laufzeit beträgt 50 s, was einer möglichen Laufzeitreduzierung von 7,5 % entspricht. Weiterhin ist zu erkennen, dass keine der gewählten Priorisierungsvarianten durchgehend zur bestmöglichen Laufzeit führt. Die Verwendung von Heuristiken ist daher nur bedingt geeignet. Größen der Parallelisierungsdomäne von $n_p = 11; 17; 19$ führen in der Simulation sogar zu einer Vergrößerung der Laufzeit. Heuristiken können für diese Problemstellung also nicht uneingeschränkt verwendet werden, da die etablierten Heuristiken in der Regel von einer konstanten Anzahl an Ressourcen ausgehen.

Die Verwendung von Optimierungsverfahren zur Ermittlung der optimalen Prozessprioritäten stellt eine Alternative zu den Heuristiken dar (s. Abschnitt 4.5). In Abbildung 5.14 ist zu erkennen, dass in der Simulation die optimierten Prioritäten immer in der minimalen Laufzeit resultieren. Die Ergebnisse liegen stets am unteren Rand des Lösungsraumes. Im Gegensatz zur allgemeinen Formulierung von Heuristiken erfolgt die Ermittlung der Prioritäten durch Optimierungsverfahren ausschließlich bezogen auf die geltenden Randbedingungen, welche hierbei durch die Anzahl der Slots n_s und die Größe der Parallelisierungsdomäne n_p repräsentiert werden. Die Größe der Parallelisierungsdomäne führt also zu unterschiedlichen Prioritätswerten der Prozesselemente, wodurch die minimale Laufzeit gewährleistet wird. Die Auswahl eines geeigneten Optimierungsverfahrens erfolgt im weiteren Verlauf dieses Abschnittes.

Die beschriebenen Sachverhalte können auch für das zweite Testszenario nachgewiesen werden, wie in Abbildung 5.15 zu sehen ist. Auch hier basieren die realen Laufzeiten jeweils auf dem Mittelwert aus fünf Durchläufen. Analog zu Testszenario 1 wird die grundlegende Charakteristik der Allokation durch das Simulationsmodell wiedergegeben und die mittlere Differenz zwischen realer und simulierter Laufzeit beträgt 18 %.

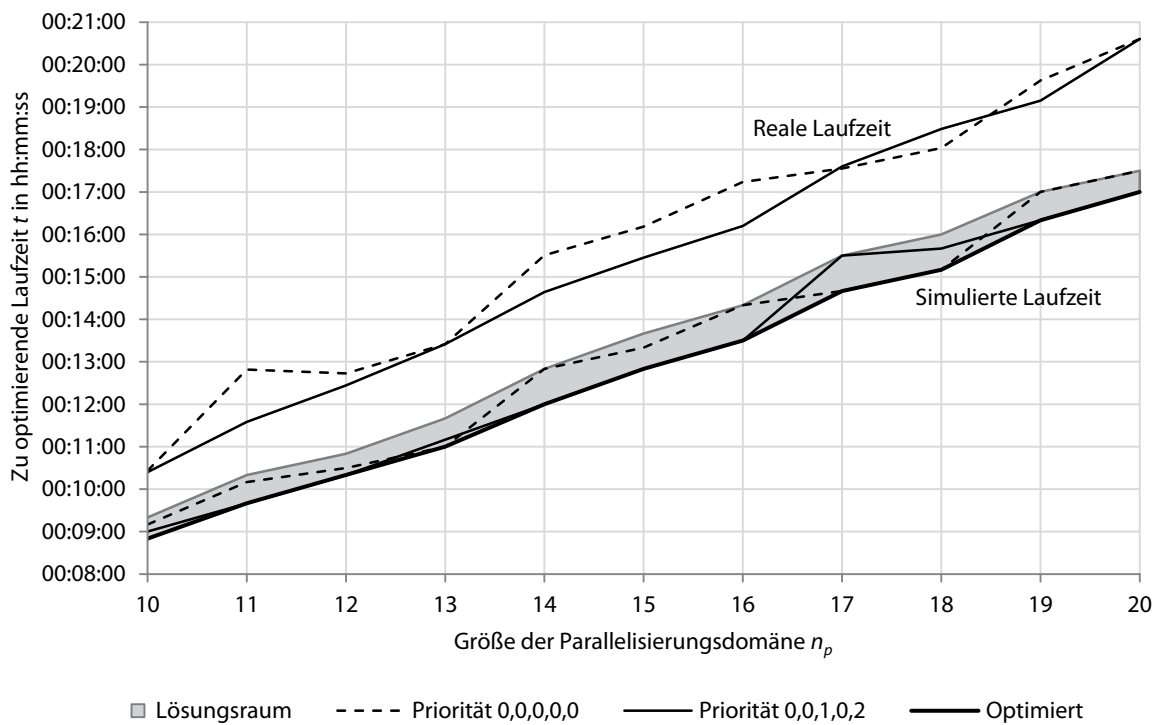


Abbildung 5.15: Vergleich von simulierter und realer Laufzeit für Testszenario 2 ($n_s = 4$)

Analog zu Testszenario 1 werden zwei Varianten von Prioritäten untersucht: die reine Verwendung von Prioritätswerten von 0, welche in der Bearbeitung der Prozesselemente nach dem FIFO-Prinzip resultiert, und die Heuristik des *kritischen Pfades*, woraus sich gemäß der Bezeichnung der Reihenfolge in Abbildung 5.13 die Prioritätswerte 0, 0, 1, 0, 2 ergeben. Auch hier führt diese Heuristik nicht durchgehend zur minimalen Laufzeit. Die minimale Laufzeit wird ausschließlich durch die optimierten Prioritätswerte erzielt. Der Einfluss der Prioritäten verläuft mit zunehmender Größe der Parallelisierungsdomäne nahezu konstant. Der maximale Einfluss zeigt sich im Bereich $n_p = 14 \dots 18$. Die Differenz zwischen der längsten und der kürzesten Laufzeit beträgt dabei 50 s. Der relative Einfluss der Laufzeitreduzierung nimmt in diesem Bereich somit ab (6,4% \rightarrow 5,2%).

In beiden Testszenarien können die optimalen Prioritäten nicht durchgängig mit den etablierten Heuristiken ermittelt werden, wenn die Größe der Parallelisierungsdomäne variiert wird. Optimierungsverfahren stellen die wesentlich flexiblere Lösung zur Ermittlung der optimalen Prioritäten dar, da bei jeder Optimierung die aktuelle Größe der Parallelisierungsdomäne sowie die zur Verfügung stehenden Ressourcen berücksichtigt werden. Aus diesem Grund wird die Verwendung eines Optimierungsverfahrens zur Ermittlung der optimalen Prioritäten weiterverfolgt.

Für die Optimierung der Prozessprioritäten stehen verschiedene Optimierungsverfahren zur Verfügung, deren Eignung im Folgenden näher untersucht wird. Daher wird eine Optimierungsstrategie definiert, bei der das Optimierungsmodell und somit die verwendeten Algorithmen einfach ausgetauscht werden können. Das Aktivitätsdiagramm für die Optimierung der Prozessprioritäten ist in Abbildung 5.16 dargestellt. Das Evaluationsmodell für die Optimierung bildet das bereits verwendete Simulationsmodell der Allokation von HTCondor (s. Aktivitätsdiagramm in Abbildung A.5 im Anhang A.6.3).

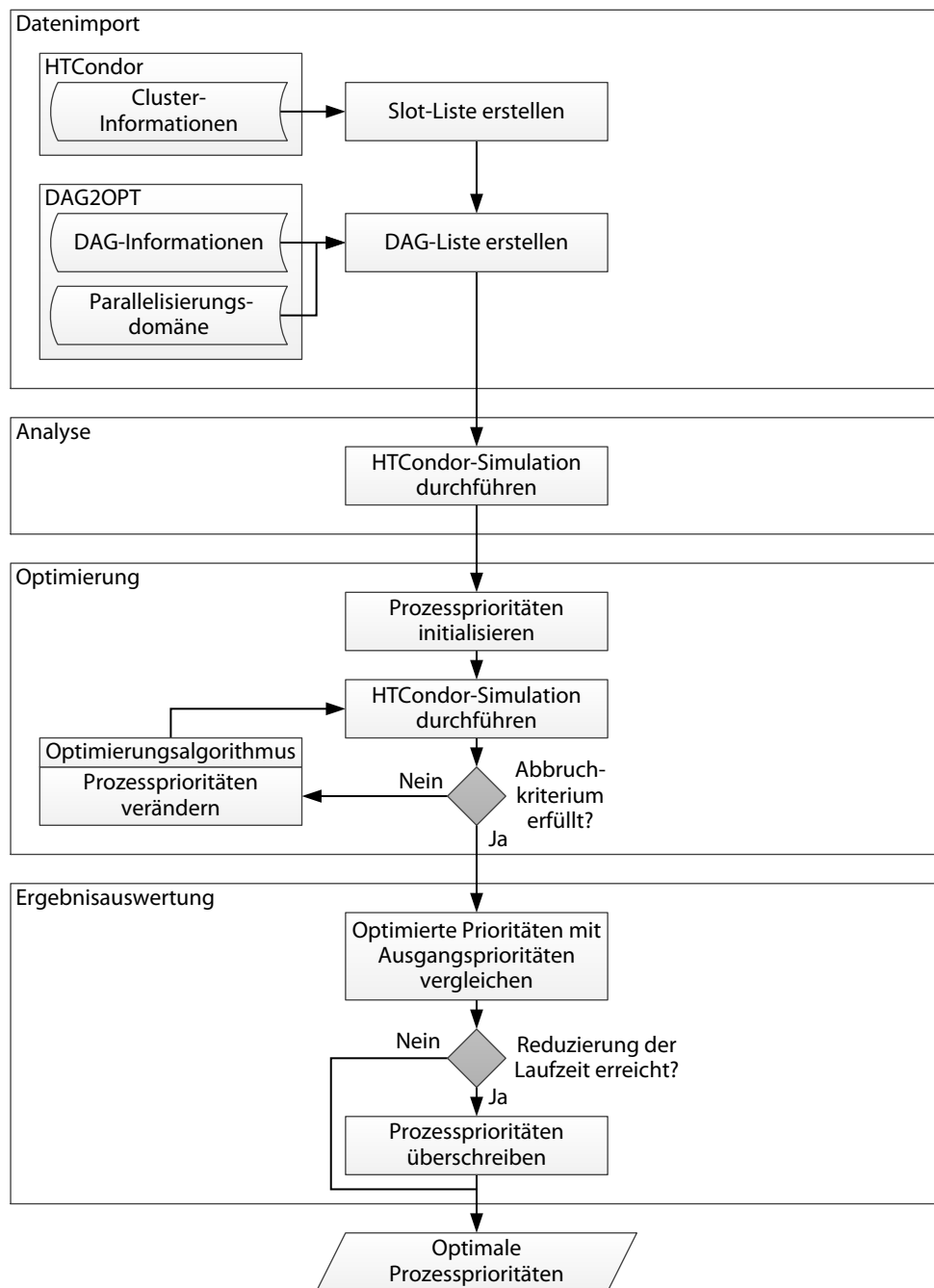


Abbildung 5.16: Aktivitätsdiagramm zur Optimierung der Prozessprioritäten

Bevor die Optimierung der Prozessprioritäten durchgeführt wird, werden analog zur reinen Allokationssimulation die benötigten Daten importiert. Anschließend erfolgt die Allokationssimulation zur Bestimmung der Laufzeit unter den aktuellen Prioritätswerten. Diese Laufzeit wird benötigt, um nach der Optimierung die Reduzierung der Laufzeit durch die optimierten Prioritäten feststellen zu können. Nach dieser Analyse der Ausgangswerte erfolgt die eigentliche Optimierung. Die Prozessprioritäten werden hierbei zunächst durch den gewählten Optimierungsalgorithmus initialisiert und variiert, bis das jeweilige Abbruchkriterium erreicht ist. Nach der Optimierung erfolgt die Auswertung der Optimierungsergebnisse. Dabei wird die optimierte Laufzeit mit der Ausgangslaufzeit verglichen. Wird durch die Optimierung eine Reduzierung der Laufzeit erzielt, werden die Ausgangsprioritäten mit den optimierten Werten überschrieben. Wird keine Laufzeitreduzierung

erzielt, stellen die Ausgangsprioritäten bereits das Optimum dar und müssen nicht überschrieben werden. Das Ergebnis der Optimierung sind in jedem Fall die, bezogen auf die Randbedingungen (Ressourcen, DAG, Parallelisierungsdomäne), optimalen Prozessprioritäten.

Da die Optimierung der Prioritäten ein kombinatorisches Optimierungsproblem darstellt, welches auf rein diskreten Designvariablen basiert, werden in den folgenden Untersuchungen verschiedene genetische Algorithmen und das simulierte Ausglühen (Simulated Annealing) betrachtet (s. Abschnitt 2.6.2). Zur Auswahl eines geeigneten Optimierungsalgorithmus werden die folgenden stochastischen Optimierungsalgorithmen untersucht:

- Nondominated Sorting Genetic Algorithm (NSGA-II): Dieser genetische Algorithmus wurde insbesondere für die Bearbeitung multikriterieller Optimierungsaufgaben entwickelt [DPAM00], [DPAM02], kann aber auch bei monokriteriellen Aufgaben eingesetzt werden. Die Fitnesszuweisung erfolgt hierbei rangbasiert. Der Algorithmus stellt den aktuellen Stand der Technik genetischer Algorithmen dar und wird in vielen kommerziellen und freien Optimierungswerkzeugen verwendet [Kan08], [SBH10].
- Einfacher Genetischer Algorithmus (GA): Dieser Algorithmus stellt eine allgemeine Formulierung eines GA ohne spezielle Einstellungen dar.
- Genetischer Algorithmus-Traveling Salesman Problem (GA-TSP): Die Einstellungen des GA sind speziell auf das Problem des Handlungsreisenden (Traveling Salesman Problem, TSP) [Men32], [Rob49], [DFJ54] angepasst. Dabei wird eine Turnierselektion mit einer Turniergröße von 5 Individuen verwendet [Ins12]. Das TSP, bei welchem die Reihenfolge bereister Städte so optimiert wird, dass die zurückgelegte Strecke minimiert wird, stellt ebenfalls ein kombinatorisches Optimierungsproblem dar und kann analog zu Scheduling-Problemen gesehen werden [GP07].
- Simulated Annealing (SA): Das simulierte Ausglühen nach [KGV83] wurde bereits in Abschnitt 2.6.2.3 beschrieben.

Die Implementierung der Optimierungsalgorithmen erfolgt mittels der Python-Bibliothek *Inspyred*, welche verschiedene Algorithmen der biologisch inspirierten Programmierung beinhaltet [Ins12].

Die Optimierungsalgorithmen werden anhand des erreichten Zielfunktionswertes, welcher die zu optimierende Laufzeit t darstellt, und der benötigten Iterationen verglichen. Um eine objektive Gegenüberstellung zu gewährleisten, müssen bei allen Algorithmen die gleichen bzw. ähnliche Einstellungen verwendet werden. Bei der Wahl der Populationsgröße der GA wird sich an Gleichung (2.13) orientiert und eine Populationsgröße von $n_I = 20$ gewählt. Zudem wird bei allen Algorithmen das gleiche Abbruchkriterium formuliert. Die Optimierung wird beendet, wenn über 10 Generationen keine weitere Reduzierung des Zielfunktionswertes erreicht wird. Dieses auf eigenen Erfahrungswerten basierende konservative Abbruchkriterium gewährleistet eine hohe Wahrscheinlichkeit das globale Optimum zu finden.

Die Untersuchung der geeigneten Optimierungsalgorithmen erfolgt für das Testszenario 2, da dieses das komplexere Testszenario bildet. Die Größe der Parallelisierungsdomäne wird auf $n_p = 4$; 15 festgelegt. Bei diesen Werten führen lediglich 4,7% der möglichen Kombinationen der Prioritätswerte zur minimalen Laufzeit, was den kleinsten prozentualen Anteil der optimalen Prioritätswerte darstellt (s. Abbildung A.6 im Anhang A.6.4).

Abbildung 5.17 zeigt die Gegenüberstellung der untersuchten Optimierungsalgorithmen bezogen auf die optimierte Laufzeit und die zur Optimierung benötigten Iterationen. Die Größe der Parallelisierungsdomäne beträgt $n_p = 4$, die Anzahl der Slots bzw. Rechner $n_s = 4$.

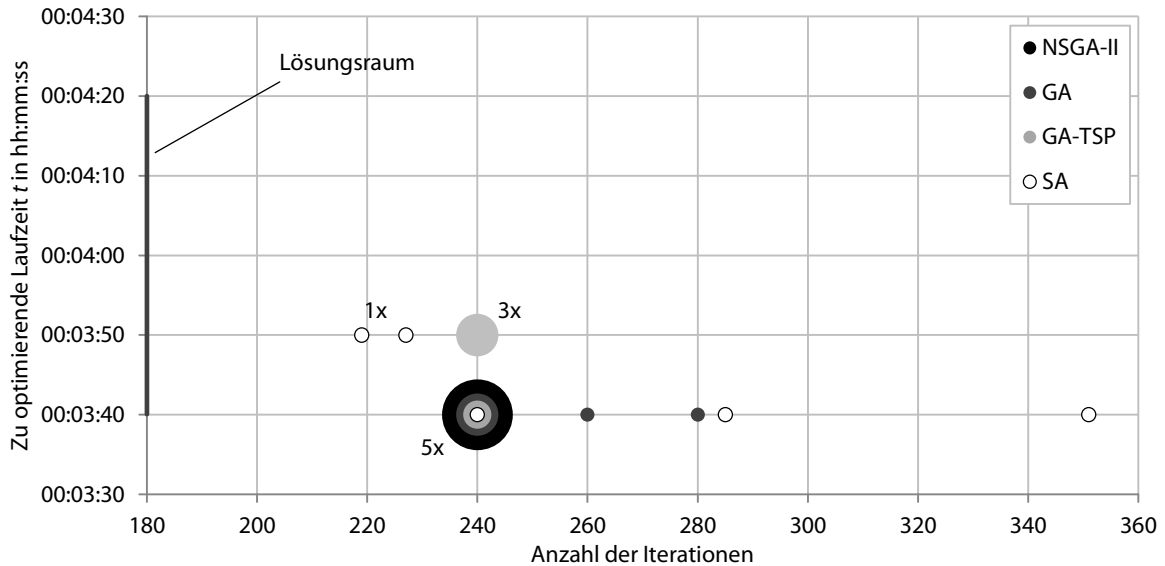


Abbildung 5.17: Vergleich verschiedener Optimierungsalgorithmen zur Optimierung der Prozessprioritäten für Testszenario 2 ($n_p = 4$, $n_s = 4$)

Da aufgrund der stochastischen Anteile der Optimierungsalgorithmen die Ergebnisse einer Streuung unterliegen, werden mit jedem Algorithmus fünf Optimierungsläufe durchgeführt. Die Größe der Kreise repräsentiert die Anzahl der Optimierungsläufe, welche den jeweiligen Punkt als Ergebnis generieren. Exemplarisch ist in dem Diagramm die jeweilige Anzahl der dazugehörigen Kreise angegeben. Weiterhin ist an der vertikalen Achse der zur Verfügung stehende Lösungsraum der Laufzeiten markiert. Da die Optimierung der Prioritäten eine monokriterielle Optimierungsaufgabe darstellt, hat auch der Lösungsraum eindimensionalen Charakter. Die optimalen Prioritäten führen zur minimalen Laufzeit, welche die untere Grenze des Lösungsraumes repräsentiert. Die maximal mögliche Laufzeit bildet die obere Grenze des Lösungsraumes.

Es ist zu erkennen, dass der Algorithmus NSGA-II in jedem Optimierungslauf die optimalen Prioritäten generiert, welche in der minimalen Laufzeit t resultieren, wobei in jedem Optimierungslauf 240 Iterationen benötigt werden. Auch der einfache GA führt stets zu den optimalen Prioritätswerten. Dabei werden dreimal 240 und jeweils einmal 260 bzw. 280 Iterationen erreicht. Durch die Algorithmen SA und GA-TSP werden nicht in jedem Optimierungslauf die optimalen Prioritätswerte gefunden. Insbesondere die Anzahl der Iterationen des SA variieren dabei sehr stark.

Ähnliche Ergebnisse zeigen sich bei einer Parallelisierungsdomäne von $n_p = 15$, welche in Abbildung 5.18 dargestellt sind.

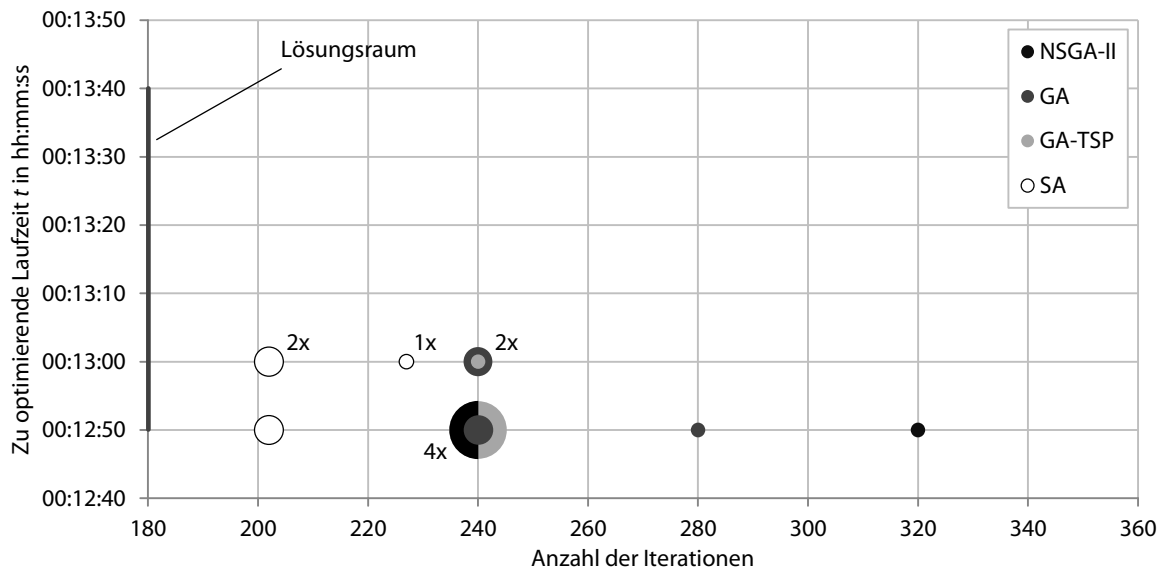


Abbildung 5.18: Vergleich verschiedener Optimierungsalgorithmen zur Optimierung der Prozessprioritäten für Testszenario 2 ($n_p = 15$, $n_s = 4$)

Hierbei werden nur durch den Algorithmus NSGA-II in jedem Optimierungslauf die optimalen Prioritätswerte ermittelt, wobei viermal 240 und in einem Durchlauf 320 Iterationen benötigt werden. Die weiteren Optimierungsalgorithmen führen nicht immer zu den optimalen Prioritätswerten und sind daher für eine weitere Verwendung zur Ermittlung der Prozessprioritäten nicht geeignet.

Die Validierung des entwickelten Werkzeugs zur Ermittlung der optimalen Prozessprioritäten erfolgt im Testszenario 3, welches das komplexeste Testszenario darstellt. Der DAG dieses Testszenarios besteht aus 8 Prozesselementen unterschiedlicher Laufzeit und Abhängigkeiten. Zusätzlich werden bei der Allokation verschiedene Software- und Lizenzrestriktionen berücksichtigt. Eine detaillierte Beschreibung und Darstellung des Testszenarios findet sich im Anhang A.6.5.

Zur Validierung werden die Prozessprioritäten mit dem entwickelten Optimierungswerkzeug ermittelt und anschließend mit diesen Prioritäten drei Durchläufe in dem realen Cluster durchgeführt. Die Ergebnisse der Zeitmessungen in dem realen Cluster sind in Abbildung 5.19 dargestellt. Zusätzlich sind die Laufzeiten bei Verwendung der neutralen Prioritäten und der ungünstigsten Prioritäten abgebildet. Die Strichlinien mit den dazugehörigen Zahlenwerten repräsentieren die jeweiligen Mittelwerte der drei Messungen. Die Größe der Parallelisierungsdomäne beträgt $n_p = 11$, die Anzahl der Slots bzw. Rechner $n_s = 4$.

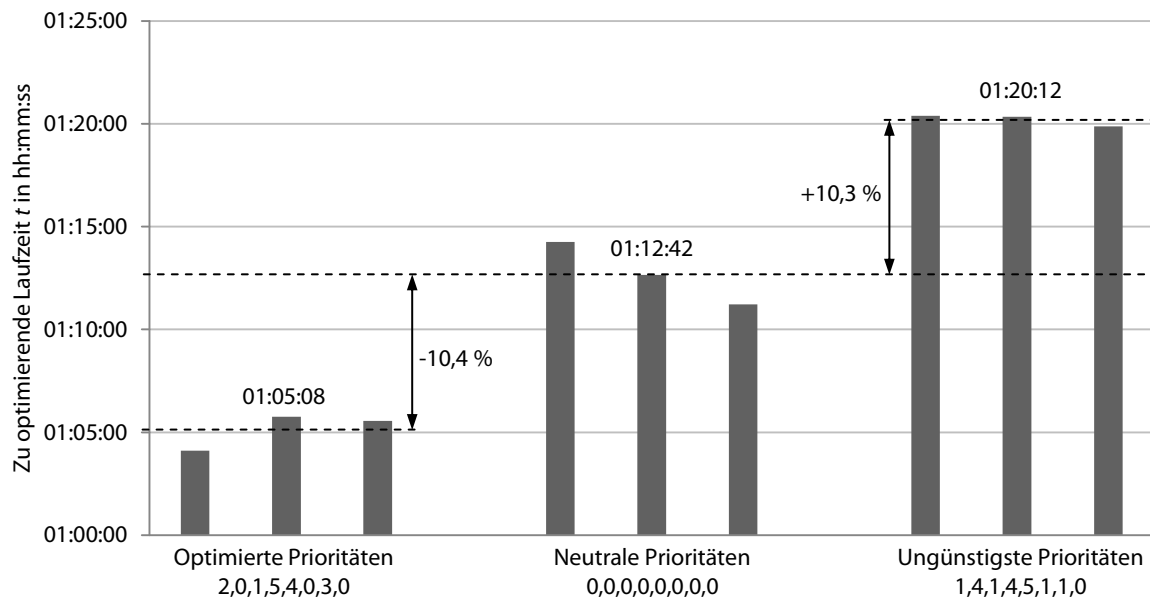


Abbildung 5.19: Laufzeitreduzierung durch Verwendung optimierter Prozessprioritäten im Testszenario 3 ($n_p = 11$, $n_s = 4$)

Die Prioritätswerte von 2, 0, 1, 5, 4, 0, 3, 0 werden durch das entwickelte Werkzeug als optimal identifiziert. Die Reihenfolge bei der Angabe der Prioritätswerte entspricht der alphabetischen Reihenfolge des in Abbildung A.7 dargestellten DAG. Durch die Optimierung der Prioritätswerte wird, ausgehend von neutraler Priorisierung, eine Laufzeitverkürzung von 7 min 34 s erreicht, was einer Reduzierung um 10,4 % entspricht. Als Gegenprobe werden die ungünstigsten Prioritätswerte ermittelt. Dazu wird die Optimierung so konfiguriert, dass die Laufzeit maximiert wird. Bei der Verwendung dieser Prioritätswerte erhöht sich die Laufzeit im Vergleich zur neutralen Priorisierung um 7 min 30 s, was einer Laufzeitverlängerung um 10,3 % entspricht.

Die Ergebnisse aus den Testszenarien zeigen die Effektivität des entwickelten Optimierungswerkzeugs. Unter Verwendung des NSGA-II Algorithmus werden die optimalen Prozessprioritäten ermittelt und dabei die jeweils zum aktuellen Zeitpunkt geltenden Randbedingungen berücksichtigt. Dies beinhaltet den verwendeten DAG, die Größe der Parallelisierungsdomäne sowie die benötigten und zur Verfügung stehenden Ressourcen.

Die Allokationssimulation bildet dabei die reale Allokation von HTCondor in ausreichender Genauigkeit ab, um im Vorfeld einer Produktoptimierung die optimalen Prioritäten der Prozesselemente bestimmen zu können. In erster Linie ist dabei die qualitative Nachbildung der Allokation von Bedeutung. Da die Allokationssimulation einen deterministischen Prozess darstellt, sind die Ergebnisse bei gleichen Prioritätswerten durchgängig reproduzierbar. Aufgrund der bereits diskutierten Schwankungen in den Messwerten bei der realen Allokation ist eine exakte quantitative Wiedergabe der realen Laufzeiten durch die Simulation schwer umsetzbar und nicht zielführend. Eine Näherung zur Vorhersage der realen Laufzeit könnte unter Verwendung der ermittelten mittleren Differenz der realen und simulierten Laufzeiten von 18 % erfolgen.

Die Ergebnisqualität der Optimierung hängt stark von der Komplexität des Optimierungsproblems ab. Damit stets eine hohe Wahrscheinlichkeit gewährleistet wird, das globale Optimum zu finden, werden die Einstellungen des NSGA-II Algorithmus in Abhängigkeit von der Problemkomplexität definiert. Die Komplexität des Optimierungsproblems

wird in erster Linie durch die Anzahl der Prozesselemente des DAG bestimmt, da hierdurch die Anzahl der Designvariablen (Prioritäten) und somit die Anzahl der möglichen Kombinationen der Prioritäten festgelegt werden. Das Abbruchkriterium des Algorithmus ist indirekt und variabel zur Problemkomplexität definiert. Die Optimierung läuft solange, bis über 10 Generationen hinweg keine Verbesserung des Fitnesswertes erreicht wird. Um in jeder Generation eine ausreichend große Population und somit einen großen Genpool zu erreichen, wird die Populationsgröße in Abhängigkeit von der Problemkomplexität definiert. Hierbei beträgt die Populationsgröße das Fünffache der Anzahl der Prozesselemente ($n_I = 5 \cdot n_e$). Die Populationsgröße richtet sich somit nach der Anzahl der möglichen Lösungen. Dies stellt im Vergleich mit der Empfehlung zur Wahl der Populationsgröße (s. Gleichung (2.13)) eine konservative und großzügige Festlegung dar. Die hieraus resultierende erhöhte Anzahl an Evaluationen kann aufgrund der kurzen Laufzeit der Allokationssimulation vernachlässigt werden. Somit wird in jedem Fall eine hohe Wahrscheinlichkeit gewährleistet, die optimalen Prioritätswerte zu ermitteln.

Zur Erhöhung der Benutzungsfreundlichkeit des Optimierungswerkzeugs wird eine Laufzeitvorhersage für die Optimierung der Prozessprioritäten implementiert. Der Anwender hat dadurch die Möglichkeit, gemäß der voraussichtlichen Optimierungslaufzeit den geeigneten Zeitpunkt für die Optimierung der Prozessprioritäten festzulegen, z. B. während einer Pause oder einer Besprechung. Die Vorhersage der zur Optimierung der Prozessprioritäten benötigten Laufzeit erfolgt auf Basis der im Vorfeld der Optimierung durchgeführten Allokationssimulation zur Analyse der aktuell verwendeten Prioritätswerte (s. Abbildung 5.16). Die Laufzeit der Allokationssimulation hängt dabei von der Größe der Parallelisierungsdomäne n_p , der Anzahl der Slots n_s sowie der Anzahl der Prozesselemente des DAG n_e ab.

Zudem ist die Laufzeit der Allokationssimulation von der Taktrate der CPU des verwendeten Rechners abhängig. Durch die einmalige Durchführung der Allokationssimulation im Vorfeld der Optimierung wird die Taktrate der CPU direkt bei der Laufzeitvorhersage berücksichtigt. Die Laufzeit der Allokationssimulation wird anschließend mit der voraussichtlichen Anzahl an Evaluationen multipliziert, welche durch die Populationsgröße ($n_I = 5 \cdot n_e$) und die Anzahl der Generationen bestimmt wird. Das Ergebnis der Laufzeitvorhersage ist die voraussichtliche Laufzeit der Optimierung der Prozessprioritäten gemäß der verwendeten Eingabeparameter. Die Abhängigkeit der Laufzeit der Allokationssimulation von den Eingabeparametern sowie die Ermittlung der mindestens benötigten Evaluationen der Optimierung sind detailliert im Anhang A.6.6 beschrieben.

Im Anschluss an die Optimierung werden dem Anwender die optimierten Prioritätswerte sowie die dadurch erreichte Laufzeitreduzierung angezeigt. Abbildung 5.20 zeigt den Dialog zur Analyse der Optimierungsergebnisse für das Testszenario 2.

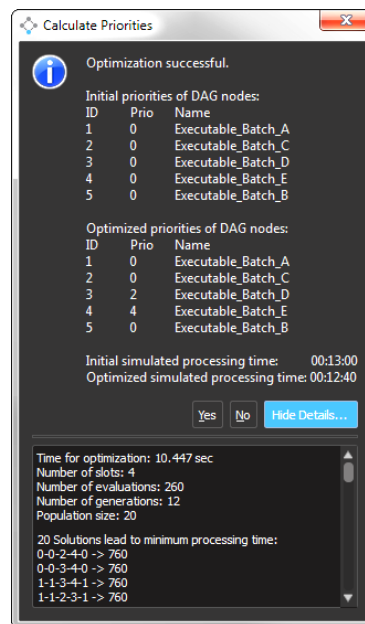


Abbildung 5.20: Ergebnis der Optimierung der Prozessprioritäten in DAG2OPT für Test-szenario 2

Weiterhin hat der Anwender über diesen Dialog Zugriff auf detaillierte Informationen der Prioritätsoptimierung. Hierzu zählen die Laufzeit der Optimierung, die Anzahl der berücksichtigten Slots, die verwendete Populationsgröße sowie die benötigte Anzahl an Generationen und Evaluationen. Zudem werden alle Prioritätswerte aufgelistet, welche zur minimalen Laufzeit führen.

5.5 Validierung durch Fallstudien

Die Validierung des entwickelten Optimierungssystems erfolgt anhand von drei Fallstudien, welche multidisziplinäre Optimierungsaufgaben aus realen Entwicklungsprojekten widerspiegeln. Besonderer Fokus bei der Validierung liegt auf dem flexiblen Einsatz des Optimierungssystems und der dynamischen Allokation der zur Verfügung stehenden Ressourcen. Dabei wird ein heterogener Workstation-Cluster in einem universitären Computer-Labor verwendet. Dieser besteht aus den folgenden Workstation-Ressourcen:

- 24 x Dell T1600 (Windows 7 64 Bit, CPU: Intel Xeon E3-1290 3,60 GHz, 16 GB RAM)
- 4 x Dell T5810 (Windows 7 64 Bit, CPU: Intel Xeon E5-1620 v3 3,50 GHz, 8 GB RAM)

Ähnliche Cluster können in vielen universitären Instituten oder Unternehmen kostenneutral aufgebaut werden, in denen eine große Anzahl durch ein Netzwerk verbundener Workstations vorhanden ist.

5.5.1 Fallstudie 1: Idealisierte Optimierung einer Interieurkomponente

Die erste Fallstudie stellt eine multidisziplinäre Optimierung einer Interieurkomponente eines Automobils dar. Zur Optimierung des Produktes wird ein GA mit einer Populationsgröße von $n_I = 50$ verwendet. Anhand dieser Fallstudie werden die in den Abschnitten 4.1 und 4.3 beschriebenen Methoden zur Parallelisierung von Optimierungsverfahren gegenübergestellt. Neben der Laufzeit werden dabei auch die benötigten Ressourcen betrachtet.

Um die Vergleichbarkeit der Parallelisierungsmethoden zu gewährleisten, werden die Evaluationsprozesse idealisiert betrachtet. Die Laufzeiten der Evaluationsprozesse der Individuen werden dabei als konstant angenommen und die Overhead-Zeiten vernachlässigt. Die Prozesskette der Optimierung unter Verwendung der parallelen Simulationsverarbeitung ist in Abbildung 5.21 dargestellt.

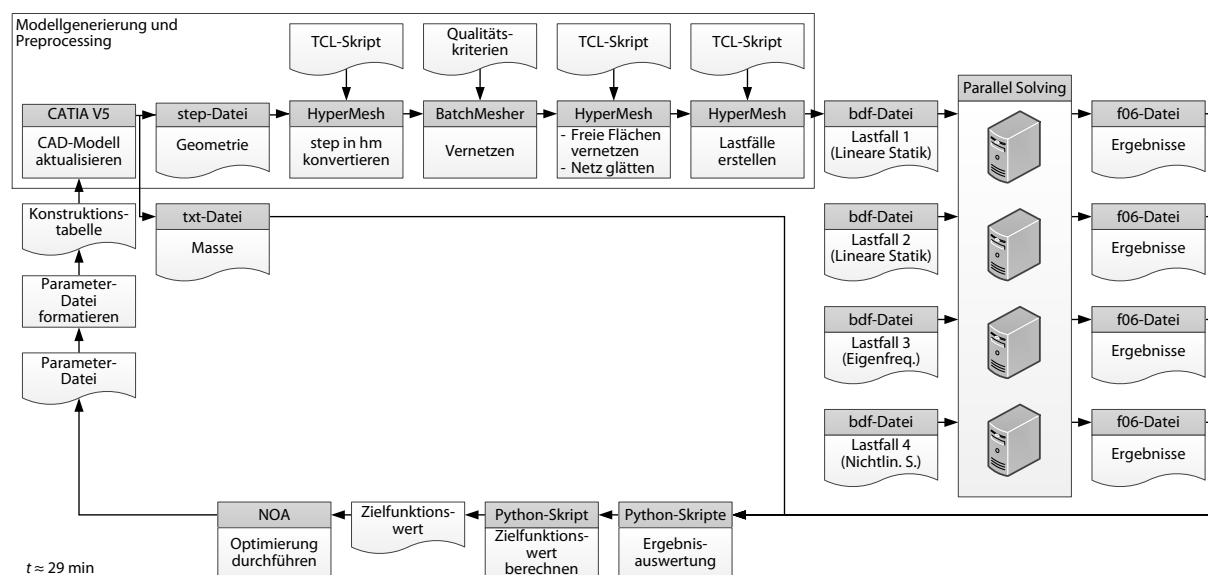


Abbildung 5.21: Prozesskette der Optimierung

Der Evaluationsprozess eines Individuums besteht aus fünf Prozesselementen. Das erste Prozesselement bildet die Modellgenerierung und das Preprocessing ab. Dies beinhaltet die Aktualisierung des CAD-Modells im CAx-System CATIA V5 sowie die Vernetzung und die Erstellung der Eingabedaten der Lastfälle für die numerische Simulation. Dabei kommen das Preprocessingtool HyperMesh und das Vernetzungstool BatchMesher zum Einsatz. Die weiteren Prozesselemente realisieren die numerische Simulation von vier Lastfällen. Die Berechnung der Lastfälle hängt von der Modellgenerierung (MG) ab und kann erst erfolgen, wenn die Eingabedaten erzeugt wurden. Die einzelnen Lastfälle sind unabhängig voneinander und können problemlos parallel bearbeitet werden (Parallel Solving).

Die Laufzeit eines Zyklus von 29 min ergibt sich bei der parallelen Simulationsverarbeitung aus der Laufzeit von Modellgenerierung und Preprocessing (9 min) und des längsten Lastfalls (Lastfall 4, 20 min). Die Laufzeiten aller Prozesselemente des Evaluationsprozesses sind in Abbildung 5.22 dargestellt.

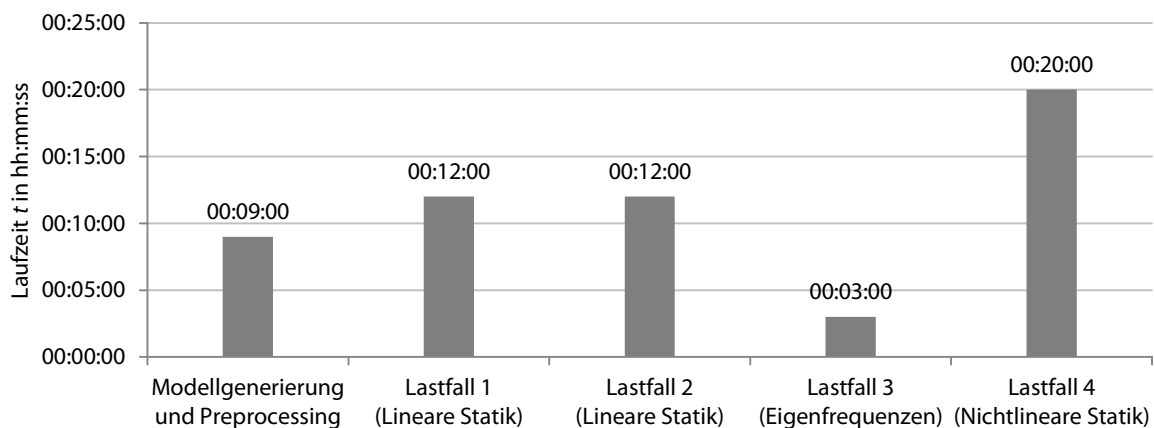


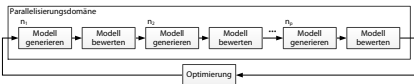
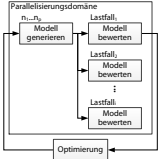
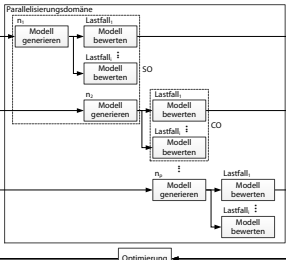
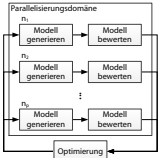
Abbildung 5.22: Laufzeiten der Prozesselemente der Evaluation, nach [WJV15]

Die Optimierung erfolgt in vier Durchläufen, in denen jeweils eine der vier beschriebenen Arten der Parallelisierung verwendet wird:

- Sequentiell (Keine Modelldekomposition)
- Parallele Simulationsverarbeitung (Dekomposition des Evaluationsmodells, Concurrent Optimization)
- Concurrent und Simultaneous Optimization (Dekomposition von Evaluations- und Optimierungsmodell)
- Vollständig parallele Verarbeitung (Dekomposition des Optimierungsmodells, Concurrent Optimization)

Bei der vollständig parallelen Verarbeitung werden mögliche Engpässe ausgeschlossen. Diese stellt somit eine idealisierte Form der Parallelisierung dar. Da bei einem GA lediglich die Evaluationsprozesse innerhalb einer Generation parallelisiert werden können, wird bei der Gegenüberstellung der Parallelisierungsmethoden die Laufzeit zur Evaluation einer Generation ausgewertet. Die Populationsgröße beträgt hierbei 50 [WJV15]. Die Laufzeiten und die bei der Evaluation verwendeten Ressourcen der vier Optimierungsläufe sind in Tabelle 5.1 dargestellt. Die in der Tabelle verwendeten Abbildungen der Parallelisierungsmethoden sind in den Abschnitten 4.1 und 4.3 in Originalgröße zu finden.

Tabelle 5.1: Laufzeiten und verwendete Ressourcen unter der Verwendung verschiedener Parallelisierungsmethoden, nach [WJV15]

Parallelisierungsmethode	Laufzeit pro Generation	Ressourcen
Sequentiell 	46:40:00	Workstations: 1 Lizenzen MG: 1 Lizenzen Sim.: 1
Parallele Simulationsverarbeitung 	24:10:00	Workstations: 5 Lizenzen MG: 1 Lizenzen Sim.: 4
Concurrent und Simultaneous Optimization 	07:50:00	Workstations: 9 Lizenzen MG: 1 Lizenzen Sim.: 8
Vollständig parallele Verarbeitung (ohne Engpässe)	00:56:00	Workstations: 50 Lizenzen MG: 50 Lizenzen Sim.: 50
		

Die vollständig parallele Verarbeitung ohne Engpässe bildet die schnellste Parallelisierungsmethode von Evaluationsprozessen einer Optimierung. Hierbei werden auch die meisten Ressourcen verwendet. Dabei führt eine große Anzahl an verwendeten Slots auch zu einer Vergrößerung des Overheads (s. Abschnitt 4.1), welcher in dieser Fallstudie vernachlässigt wird. Diese Form der Parallelisierung stellt hier ein theoretisches Beispiel der idealen angestrebten Situation dar.

Für den Vergleich der weiteren Parallelisierungsmethoden wird die Lizenz zur Modellgenerierung auf 1 beschränkt, da für den BatchMesher nur eine Lizenz zur Verfügung steht. Weiterhin erfordert der automatisierte Aufruf von HyperMesh zur Erstellung der Lastfälle den Aufruf der Benutzungsoberfläche von HyperMesh und somit einen angemeldeten Benutzer an dem verwendeten Rechner. Durch diese Beschränkungen kann dieses Prozesselement nur an einem Rechner und nicht parallel ausgeführt werden.

Die Optimierung nach dem Simultaneous Optimization-Ansatz arbeitet hier sehr effizient hinsichtlich der Verwendung der Ressourcen. Neben der Lizenz für die Modellgenerierung

werden lediglich 8 Lizenzen für die Berechnung der Lastfälle verwendet. Daraus ergeben sich 9 eingesetzte Workstations. Im Vergleich zur parallelen Simulationsverarbeitung wird durch die doppelte Anzahl an Lizenzen zur Simulation und die nahezu doppelte Anzahl an Workstations die Laufzeit um 16 h 20 min verkürzt, was einer Reduzierung um 67,6 % entspricht. Wird die Anzahl der eingeschränkten Lizenzen für die Modellgenerierung erhöht, so würden sich die Laufzeit und die Anzahl der Simulationslizenzen und Workstations der vollständig parallelen Verarbeitung angleichen. Somit wird bei der Parallelisierung der Evaluationsprozesse stets der Idealzustand angestrebt.

Das wesentliche Risiko bei der vollständig parallelen Verarbeitung ist das spontane Auftreten von Engpässen. Stehen durch eine spontane Veränderung des Clusters weniger Ressourcen als die Populationsgröße oder eines ganzzahligen Teils der Populationsgröße zur Verfügung (z. B. durch den Ausfall eines Rechners oder die Verwendung einer benötigten Lizenz durch einen anderen Anwender) entsteht Leerlauf in den Ressourcen und die Optimierung arbeitet nicht mehr effizient. Das entwickelte Framework zur Concurrent und Simultaneous Optimization hingegen kann durch eine Umverteilung der Ressourcen spontan auf die neue Situation reagieren. Dies gilt ebenfalls für die interaktive Nutzung der Workstations.

5.5.2 Fallstudie 2: DoE-Studie eines Kurbelarms

Die zweite Fallstudie stellt eine DoE-Studie des in [Wün15] simulierten und optimierten Kurbelarms dar. Dabei werden die vollständig parallele Verarbeitung und Simultaneous Optimization miteinander verglichen und die Anzahl der verwendeten Slots des Clusters variiert ($1 \leq n_s \leq 16$). Die Ergebnisse werden einer idealisierten analytischen Beschreibung (ohne Overhead) gegenübergestellt und darauf aufbauend eine Entscheidungshilfe für zukünftige ähnliche Optimierungsaufgaben gegeben.

In der Fallstudie wird ein Versuchsplan nach Plackett-Burman verwendet (s. Abschnitt 2.7.1). Da in diesem Versuchsplan die zu berechnenden Stützstellen nach einem festen Schema generiert werden, wird sichergestellt, dass in jedem Durchlauf der DoE-Studie die gleichen Designvariablenwerte berechnet werden. Dadurch wird eine Streuung der Laufzeiten der Prozesselemente durch unterschiedliche Modellgrößen ausgeschlossen. Zudem werden in diesem Versuchsplan bei sieben Designvariablen lediglich acht Stützstellen benötigt, welche die Größe der Parallelisierungsdomäne definieren ($n_p = 8$). Weiterhin werden ausschließlich die Workstations Dell T1600 und somit ein homogener Cluster verwendet. Dies reduziert ebenfalls die Streuung der Laufzeiten, da die Slots weitestgehend identisch sind. Da die Laufzeiten aufgrund nicht vorhersehbarer externer Prozesse auf den Workstations dennoch einer Streuung unterliegen, stellen sämtliche in dieser Fallstudie gemessene Laufzeiten den Mittelwert aus fünf Messungen dar.

Der Evaluationsprozess der Fallstudie besteht aus drei Prozesselementen: Modellgenerierung und Preprocessing sowie je einem Lastfall für die lineare Statik und für die Berechnung der Eigenfrequenzen. Der DAG des Evaluationsprozesses mit den jeweiligen gemittelten Laufzeiten der Prozesselemente ist in Abbildung 5.23 dargestellt.

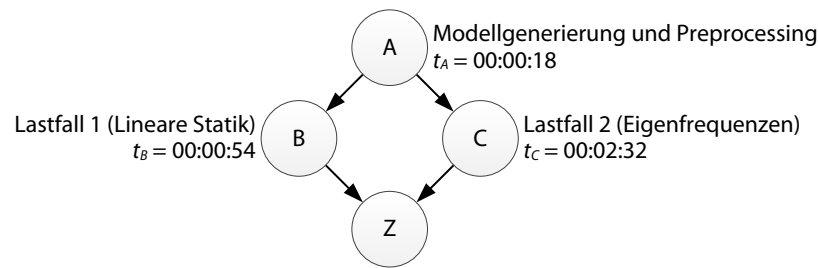


Abbildung 5.23: DAG des Evaluationsprozesses der Fallstudie 2

Die Modellgenerierung und das Preprocessing (A) erfolgt im CAx-System NX 10. Dabei wird das parametrische CAD-Modell nach dem Einlesen der Designvariablenwerte aktualisiert, die Masse des Modells extrahiert und die Eingabedateien für die Berechnung der Lastfälle generiert. Anschließend werden die beiden Lastfälle mittels des FE-Solvers NX Nastran berechnet (B, C). Die beiden Lastfälle sind unabhängig voneinander und können problemlos parallelisiert werden. Im Anschluss an die Analyse der Lastfälle erfolgt die Berechnung des Zielfunktionswertes (Z). Wie schon bei den in Abschnitt 5.4.2 verwendeten Testszenarien wird der Zielfunktionswert sehr schnell berechnet (deutlich unter 1 s). Daher kann dieses Prozesselement für die folgenden Untersuchungen vernachlässigt werden.

Die Bearbeitung des Prozesselementes A stellt keine besonderen Anforderungen an die Hardware eines Rechners und erfolgt im Vergleich zu den anderen Prozesselementen sehr schnell (18 s). NX wird dabei lediglich im Batch-Modus ohne Benutzungsoberfläche gestartet und ein Skript zum automatischen Ablauf der beschriebenen Schritte durchgeführt. Numerisch aufwendige Berechnungen werden nicht durchgeführt. Dieses Prozesselement kann somit auch auf dem Master-Rechner (s. Abschnitt 4.1) bearbeitet werden, welcher die Optimierung durchführt und die Übermittlung der Prozesselemente an den Cluster übernimmt. Die Berechnung der Lastfälle ist numerisch aufwendiger und wird ausschließlich auf den Rechnern des Clusters durchgeführt. Die beschriebene Vorgehensweise bildet den Ansatz der Simultaneous Optimization ab. Die Bearbeitung des Prozesselementes A erfolgt sequentiell und lokal. Sobald die für die Prozesselemente B und C benötigten Eingabedaten erzeugt wurden, werden diese Prozesselemente zur parallelen Bearbeitung an den Cluster übermittelt.

Im Vergleich zur vollständig parallelen Verarbeitung, bei der alle Prozesselemente parallel in dem Cluster bearbeitet werden, kann somit die Laufzeit zur Durchführung der DoE-Studie reduziert werden. Die Laufzeit t der DoE-Studie unter Verwendung beider Parallelisierungsmethoden ist in Abhängigkeit von der Anzahl der verwendeten Slots n_s des Clusters in Abbildung 5.24 dargestellt.

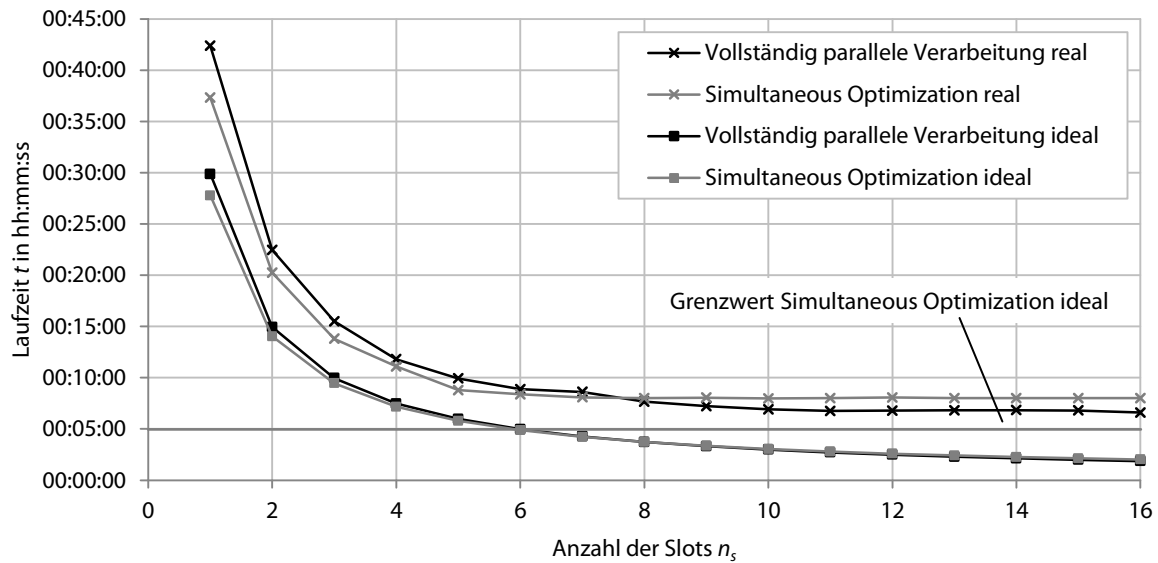


Abbildung 5.24: Reale und idealisierte Laufzeit verschiedener Parallelisierungsmethoden

Bei dem Vergleich der realen Laufzeiten der vollständig parallelen Verarbeitung und Simultaneous Optimization ist zu erkennen, dass bis zu einer Slotanzahl von $n_s = 7$ Simultaneous Optimization zu geringeren Laufzeiten führt. Dies ist zum einen mit der zusätzlichen Ressource des Master-Rechners zu erklären, welcher die Bearbeitung des Prozesselementes A übernimmt. Zum anderen wird der Overhead reduziert, da der Allokationsprozess des Prozesselementes A in dem Cluster wegfällt und die Ein- und Ausgabedaten für die Modellgenerierung und das Preprocessing nicht vom Master-Rechner an den Worker-Rechner des Clusters übermittelt werden müssen. Ab einer Slotanzahl von $n_s = 8$ führt die vollständig parallele Verarbeitung zu niedrigeren Laufzeiten, da jedem Prozesselement A der Parallelisierungsdomäne eine eigene Ressource zugewiesen wird und somit alle Prozesselemente parallel bearbeitet werden ($n_s = n_p$). Durch eine weitere Erhöhung der Slotanzahl ist nach diesem Punkt nur noch eine geringe Reduzierung der Laufzeit möglich.

Ähnliches Verhalten zeigt sich bei dem Vergleich der idealisierten Laufzeiten der beiden Parallelisierungsmethoden. Diese Laufzeiten basieren auf einer rein analytischen Beschreibung der jeweiligen Parallelisierungsmethode, in der der Overhead vernachlässigt wird. Die analytische Beschreibung der Parallelisierung wird im weiteren Verlauf dieses Abschnittes noch näher erläutert. Bis zu dem Punkt, in dem die Slotanzahl der Größe der Parallelisierungsdomäne entspricht ($n_s = n_p = 8$), werden durch Simultaneous Optimization niedrige Laufzeiten erreicht. Da bei den idealisierten Laufzeiten der Overhead vernachlässigt wird, ist die Differenz in den Laufzeiten rein auf die Verwendung des Master-Rechners als zusätzliche Ressource zurückzuführen. Weiterhin muss bei der Simultaneous Optimization der Grenzwert berücksichtigt werden, in dem die Gesamtlaufzeit durch die sequentiellen Prozesselemente limitiert wird. Dieser Grenzwert beträgt für diese Fallstudie 4 min 56 s. Schneller als dieser Wert kann die DoE-Studie mittels Simultaneous Optimization nicht ablaufen. Somit ergibt sich, dass für das idealisierte System bereits ab einer Slotanzahl von $n_s = 7$ die vollständig parallele Verarbeitung schneller abläuft.

Weiterhin ist in dem Diagramm zu erkennen, dass die idealisierten Laufzeiten gut durch die realen Laufzeiten abgebildet werden. Beide Kurven verlaufen annähernd parallel zueinander. Die mittlere Differenz zwischen den idealisierten und realen Kurven beträgt 282 s. Diese Differenz kann mit der experimentell bestimmten mittleren Overhead-Zeit

eines Jobs von 10 s erklärt werden (s. Anhang A.6.1). So müssen bei der vollständig parallelen Verarbeitung 24 Jobs den Ressourcen zugewiesen werden. Diese Zahl ergibt sich aus den 3 Prozesselementen multipliziert mit der Größe der Parallelisierungsdomäne von $n_p = 8$. Daraus resultiert eine Overhead-Zeit von 240 s. Die Abweichung von 42 s ist auf Schwankungen in den realen Overhead-Zeiten zurückzuführen. Zudem ist in dem Diagramm erkennbar, dass die Overhead-Zeit (Differenz zwischen idealisierten und realen Kurven) mit zunehmender Slotanzahl geringer wird, da mehr Ressourcen zur Verfügung stehen und der Allokationsprozess von HTCondor schneller durchgeführt wird.

Die idealisierten Laufzeiten beider Parallelisierungsmethoden basieren auf rein analytischen Beschreibungen. Dabei sind die Größe der Parallelisierungsdomäne n_p , die Anzahl der Slots n_s sowie die Laufzeiten der parallelen Prozesselemente t_{p_i} und der sequentiellen Prozesselemente t_{seq_i} maßgebend. Die idealisierten Laufzeiten werden durch die folgenden Gleichungen beschrieben:

$$\text{Vollständig parallele Verarbeitung ideal:} \quad t = \frac{n_p}{n_s} \cdot \sum t_{p_i} \quad (5.1)$$

$$\text{Simultaneous Optimization ideal:} \quad t = \sum t_{seq_i} + \frac{n_p}{n_s} \cdot \sum t_{p_i} \quad (5.2)$$

$$\text{Grenzwert Simultaneous Optimization ideal:} \quad t = n_p \cdot \sum t_{seq_i} + t_{p_{max}} \quad (5.3)$$

Da bei der vollständig parallelen Verarbeitung alle Prozesselemente parallel bearbeitet werden, ergibt sich die idealisierte Laufzeit aus dem Produkt der aufsummierten Laufzeiten der Prozesselemente t_{p_i} und dem Quotient aus Größe der Parallelisierungsdomäne n_p und Anzahl der Slots n_s . Bei Simultaneous Optimization reduziert sich die Summe der parallelen Laufzeiten, da einige Prozesselemente sequentiell bearbeitet werden. Diese Laufzeit der sequentiellen Prozesselemente t_{seq_i} wird zur parallelen Laufzeit addiert. Zusätzlich muss der Grenzwert der Simultaneous Optimization berechnet werden. Dieser ergibt sich aus der Größe der Parallelisierungsdomäne n_p und den aufsummierten sequentiellen Prozesselementen t_{seq_i} . Hierzu muss die Laufzeit des längsten parallelen Prozesselementes $t_{p_{max}}$ addiert werden.

Bezogen auf diese Fallstudie ergeben sich die folgenden Gleichungen, durch welche die in Abbildung 5.24 dargestellten Kurven beschrieben werden:

$$\text{Vollständig parallele Verarbeitung ideal:} \quad t = \frac{8}{n_s} \cdot (t_A + t_B + t_C) \quad (5.4)$$

$$\text{Simultaneous Optimization ideal:} \quad t = t_A + \frac{8}{n_s} \cdot (t_B + t_C) \quad (5.5)$$

$$\begin{aligned} \text{Grenzwert Simultaneous Optimization ideal:} \quad t &= 8 \cdot t_A + t_C & (5.6) \\ &= 296 \text{ s} \hat{=} 00:04:56 \end{aligned}$$

Zur Bewertung des Parallelisierungsgrades wird aus den Laufzeiten der vollständig parallelen Verarbeitung der durch die Parallelisierung erreichte Speedup S_p und die Effizienz E_p berechnet (s. Abschnitt 4.1). Die idealisierten Laufzeiten der vollständig parallelen Verarbeitung dienen dabei als Referenz für den Idealzustand. Hieraus kann der lineare Speedup und die ideale Effizienz abgeleitet werden. Da bei Simultaneous Optimization der Master-Rechner als zusätzliche Ressource verwendet wird und dies streng genommen eine heterogene Umgebung darstellt, da auf dieser Ressource die Prozesselemente B und C nicht bearbeitet werden können, wird auf die Darstellung des Speedup und der Effi-

zienz von Simultaneous Optimization verzichtet. Abbildung 5.25 zeigt den Speedup und die Effizienz der realen und idealisierten Laufzeiten in Abhängigkeit von der Anzahl der Slots n_s .

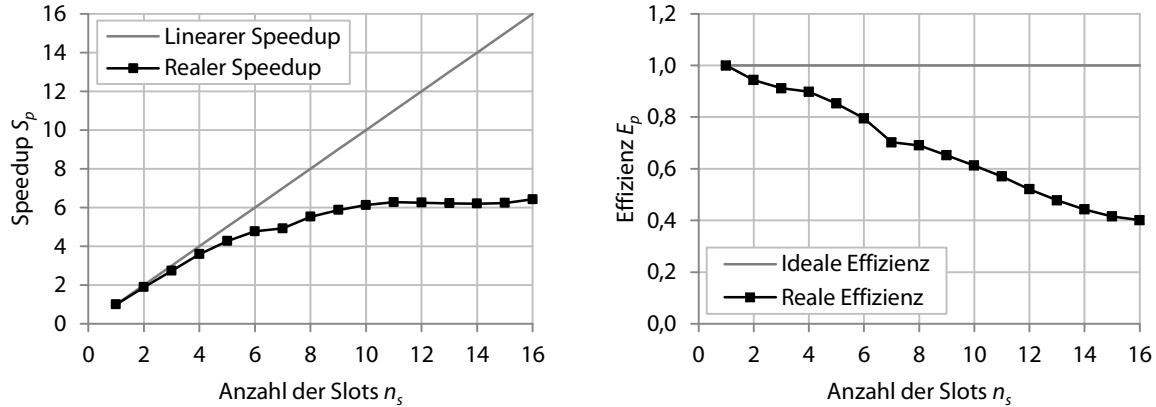


Abbildung 5.25: Speedup und Effizienz der vollständig parallelen Verarbeitung

In den Diagrammen zeigt sich deutlich der Einfluss des Overheads. Obwohl die absoluten Overhead-Zeiten mit zunehmender Anzahl der Slots annähernd konstant bleiben bzw. leicht reduziert werden (s. Abbildung 5.24), erhöht sich aufgrund der geringer werdenden absoluten Laufzeit t der relative Einfluss der Overhead-Zeiten. Dies führt mit zunehmender Slotanzahl n_s zu einem geringeren Anstieg des Speedup, was in einer geringer werdenden Effizienz resultiert. Die Effizienz reduziert sich hierbei annähernd linear.

Bei einer im Vergleich zur Größe der Parallelisierungsdomäne geringen Anzahl an Slots ($n_s \leq n_p/2 = 4$) wird durch die vollständig parallele Verarbeitung annähernd linearer Speedup erreicht, was einer Effizienz von $E_p \geq 0,9$ entspricht. Auch eine Slotanzahl von $n_s = 6$ führt noch zu einer sehr guten Effizienz von $E_p = 0,8$. Mit zunehmender Slotanzahl nimmt die Effizienz der Parallelisierung ab. Ab einer Slotanzahl von $n_s \geq 10$ bleibt der weitgehend konstant bzw. wird nur noch sehr wenig gesteigert.

Im Vorfeld einer Optimierung oder einer DoE-Studie ist es in jedem Fall zu empfehlen, die sequentielle Bearbeitung einiger Prozesselemente zu überprüfen (z. B. am Anfang bzw. am Ende des DAG). Insbesondere wenn nur eine begrenzte Anzahl von Ressourcen zur Verfügung steht und einige Prozesselemente numerisch nicht sehr aufwendig sind, kann durch Simultaneous Optimization und die Nutzung des Master-Rechners als zusätzliche Ressource die Laufzeit reduziert werden. Als Entscheidungshilfe hierzu dienen die idealisierten Laufzeiten beider Parallelisierungsmethoden sowie der Grenzwert der Simultaneous Optimization (s. Gleichungen (5.1) – (5.3)).

Neben den Prozesselementen ist das Verhältnis n_p/n_s entscheidend für einen hohen Wert von Speedup S_p und Effizienz E_p . Ist die Anzahl der Slots n_s bekannt und die Größe der Parallelisierungsdomäne n_p frei wählbar (z. B. die Anzahl der Stützstellen eines Latin-Hypercube-Versuchsplans), kann durch eine größere Parallelisierungsdomäne ($n_p > n_s$) eine hohe Effizienz der zur Verfügung stehenden Ressourcen erzielt werden.

5.5.3 Fallstudie 3: Optimierung eines Hebelmechanismus

Die dritte Fallstudie beinhaltet die multidisziplinäre Optimierung eines Hebelmechanismus unter Verwendung eines GA. Dabei werden alle zur Verfügung stehenden Ressourcen des Clusters über eine Dauer von sieben Tagen genutzt. Somit wird die vollständige heterogene Cluster-Umgebung verwendet. Parallel zur durchgeführten Optimierung werden die Rechner des Clusters zur interaktiven Arbeit genutzt. Dies erfolgt spontan sowie periodisch während Lehrveranstaltungen.

Die Struktur des bei der Optimierung verwendeten Evaluationsprozesses ist analog zum Evaluationsprozess in Fallstudie 2 (s. Abschnitt 5.5.2). Neben der Modellgenerierung und dem Preprocessing erfolgt im Prozesselement A ein Export des Geometriemodells als step-Datei, welche zur Archivierung der evaluierten Varianten genutzt wird. Dieser Export führt zu einer starken Streuung der Laufzeit t . Hinzu kommt die Streuung aufgrund der unterschiedlichen Modellgröße der zu evaluierenden Varianten, welche alle Prozesselemente betrifft. Die Laufzeit der Prozesselemente wird daher über den Mittelwert aus 230 zufällig generierten Varianten berechnet. Daraus ergeben sich folgende Laufzeiten:

- Prozesselement A: Modellgenerierung und Preprocessing, $t_A = 00:01:35$
- Prozesselement B: Lastfall 1 (Lineare Statik), $t_B = 00:02:51$
- Prozesselement C: Lastfall 2 (Eigenfrequenzen), $t_C = 00:09:49$

Die mittlere Gesamtlaufzeit des Evaluationsprozesses beträgt $t = 00:14:15$. Eine detaillierte Beschreibung des Evaluationsprozesses sowie eine Übersicht der Laufzeitverteilung der Prozesselemente ist im Anhang A.7.1 zu finden.

Die Fallstudie beinhaltet zudem die Verwendung von Anforderungen der Prozesselemente an die Hardware der Rechner (s. Abschnitt 5.3.6). Da der Arbeitsspeicher der Dell T5810 Workstations von 8 GB RAM zur Bearbeitung der Prozesselemente B und C nicht ausreichend ist, wird eine Mindestanforderung von 12 GB RAM definiert. Somit werden diese Prozesselemente ausschließlich auf den leistungsstärkeren Workstations bearbeitet.

Bei der Optimierung des Hebelmechanismus werden zwei Optimierungsläufe betrachtet, welche teilweise überlappend und ebenfalls parallel in dem Cluster durchgeführt wurden. Beide Optimierungsläufe unterscheiden sich in dem parametrischen CAD-Modell. Das Modell der ersten Optimierung besitzt an der Unterseite des Klemmhebels vier kreuzförmige Rippenstrukturen. Das Modell der zweiten Optimierung besitzt drei kreuzförmige Rippenstrukturen (s. Abbildung A.19 im Anhang A.7.2). Somit kann dieses Modell Produktvarianten mit einer geringeren Masse generieren, wie in der Gegenüberstellung der Zielkriterien m und u_{max} in Abbildung 5.26 dargestellt ist.

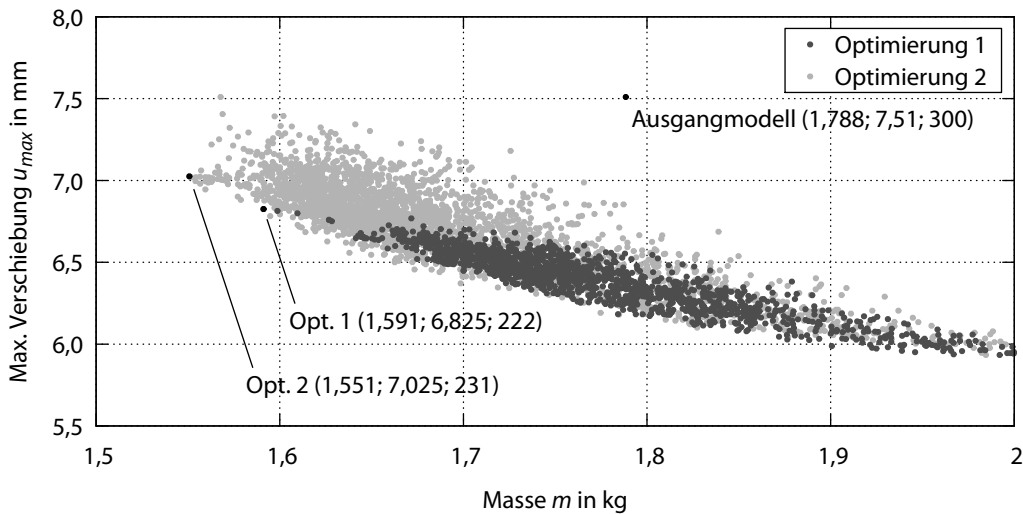


Abbildung 5.26: Vergleich der Ergebnisse der durchgeführten Optimierungen

Die evaluierten Individuen beider Optimierungsläufe führen im dargestellten Lösungsraum zu einer deutlichen Ausprägung der Pareto-Front. Im Vergleich zum Ausgangsmodell wird durch beide Optimierungen eine Reduzierung der Werte der Zielkriterien erreicht. Die in der Abbildung markierten Individuen spiegeln jeweils den geringsten Zielfunktionswert wider. Die in Klammern gesetzten Werte stellen die Werte der Zielkriterien m , u_{max} und σ_{max} dar. Eine detaillierte Beschreibung der Zielkriterien und der Optimierungsergebnisse ist im Anhang A.7.2 zu finden.

Beide Optimierungen wurden zeitversetzt durchgeführt. Die Anpassung des parametrischen CAD-Modells erfolgte, nachdem anhand von Zwischenergebnissen erkennbar war, dass das Entfernen einer kreuzförmigen Rippenstruktur weiteres Potential zur Reduzierung der Masse bietet. Die zweite Optimierung wurde nach ca. 40% der insgesamt benötigten Evaluationen des ersten Optimierungslaufs gestartet. Der erste Optimierungslauf beinhaltete 1553, der zweite 2966 Evaluationen. Die zeitliche Verteilung der durchgeführten Evaluationen beider Optimierungen ist in Abbildung 5.27 dargestellt.

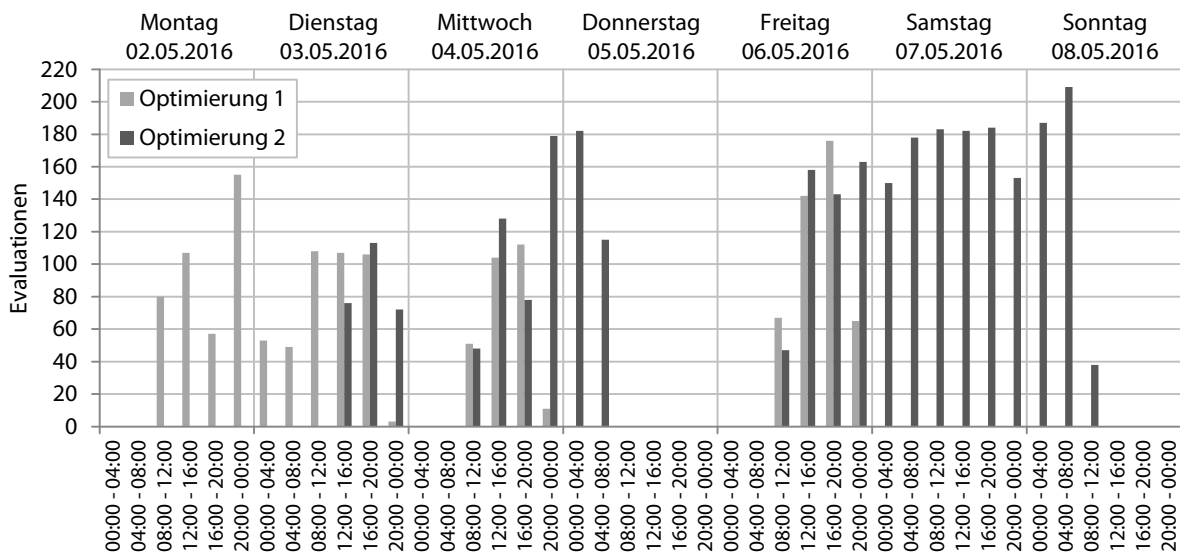


Abbildung 5.27: Evaluationen der Optimierungen

Anhand der Schwankungen der Evaluationen in den ersten drei Tagen ist die interaktive Nutzung der Rechner des Clusters aufgrund von Lehrveranstaltungen und sonstigen Tätigkeiten zu erkennen. So wurden in den an diese Phasen anschließenden Zeiträumen 20:00 – 00:00 am 02.05.2016 und 20:00 – 00:00 am 04.05.2016 deutlich mehr Evaluationen durchgeführt als in den Zeiträumen davor. An den letzten drei Tagen fanden keine Lehrveranstaltungen statt, wodurch alle Rechner des Clusters für die Optimierung verwendet wurden. Schwankungen in der Anzahl der Evaluationen in diesem Zeitraum sind auf die bereits beschriebene Streuung der Laufzeiten der Evaluationsprozesse zurückzuführen.

Zudem sind die durch anfängliche Fehler im Optimierungssystem DAG2OPT und in der Konfiguration des Clusters hervorgerufenen Ausfälle einzelner Ressourcen sowie des gesamten Clusters zu erkennen (00:00 – 04:00 am 03.05.2016, 00:00 – 04:00 am 04.05.2016, 08:00 – 12:00 am 05.05.2016). Nach der Behebung der Fehler wurde die Optimierung an den letzten drei Tagen stabil durchgeführt.

Auch die parallele Bearbeitung von zwei Optimierungen in dem Cluster stellt kein Problem dar. So wurden im Zeitraum 16:00 – 20:00 am 06.05.2016 aus der ersten Optimierung 176 und aus der zweiten 143 Evaluationen parallel durchgeführt. Dies wird zudem durch die verwendeten Operationen zwischen den einzelnen Generationen unterstützt. So wird bei der Optimierung zwischen jeder Generation das beste Individuum zweimal lokal modifiziert und das schlechteste Individuum der Generation durch ein zufällig generiertes ersetzt. Diese Evaluationen sind innerhalb von NOA nicht parallelisiert und werden seriell ausgeführt. Lediglich die Berechnung der Lastfälle (Prozesselemente B und C) erfolgt parallel. Während dieser Operationen stehen die Rechner des Cluster somit für die jeweils andere Optimierung zur Verfügung.

Zur Ermittlung des erreichten Parallelisierungsgrades wurde die zweite Optimierung erneut durchgeführt. Auf die oben beschriebenen Operationen zwischen den Generationen wurde verzichtet, um sequentielle Anteile der Optimierung auszuschließen. Dieser Optimierungslauf führte somit zu 2828 Evaluationen. Die zeitliche Verteilung der durchgeführten Evaluationen beider Optimierungen ist in Abbildung 5.28 dargestellt.

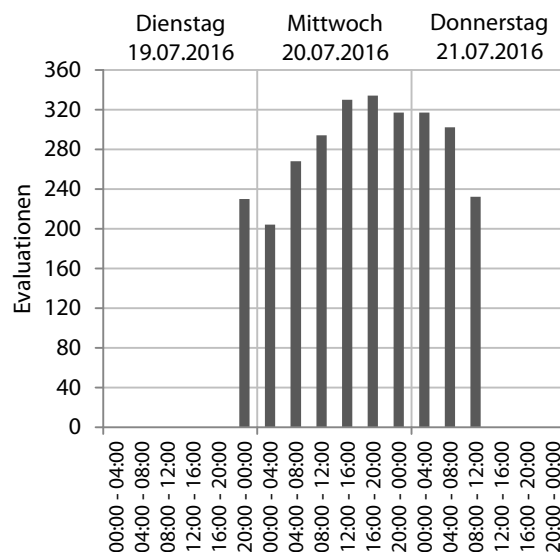


Abbildung 5.28: Evaluationen von Optimierung 2 bei einer erneuten Durchführung

Auch bei diesem Optimierungslauf standen die Rechner des Clusters jederzeit für interaktive Arbeit zur Verfügung, was neben den streuenden Laufzeiten zu Schwankungen der Evaluationen führte. Lehrveranstaltungen wurden während der Optimierung nicht durchgeführt. Zudem erfolgte im Zeitraum 00:00 – 04:00 am 20.07.2016 die Installation eines Updates des Betriebssystems, währenddessen die Rechner des Clusters nicht zur Bearbeitung der Prozesselemente bereitstanden und somit weniger Evaluationen durchgeführt wurden.

Bei der Optimierung wurden 2828 Evaluationen durchgeführt. Multipliziert mit der mittleren Laufzeit einer Evaluation ergibt dies bei einer reinen sequentiellen Bearbeitung eine Gesamtlaufzeit von 671 h 39 min ($\cong 27,99$ d). Diese Laufzeit ist für die Durchführung einer solchen Optimierung deutlich zu lang. Durch die Parallelisierung der Optimierung kann die Gesamtlaufzeit auf 38 h 35 min ($\cong 1,61$ d) verkürzt werden.

Aus diesen Laufzeiten ergibt sich für die gesamte Optimierung ein Speedup von $S_p = 17,4$. Da die bei der Optimierung verwendeten Rechner eine heterogene Umgebung darstellen, in der zudem nicht alle Prozesselemente auf den gleichen Rechnern bearbeitet werden können, ist die Verwendung der gesamten Rechneranzahl zur Berechnung des Effizienzwertes nicht geeignet. Die vier Dell T5810 Workstations werden durch die Bearbeitung der Prozesselemente (A) nur maximal zur Hälfte ausgelastet. Daher wird die Anzahl dieser Rechner halbiert und zu den 23 Dell T1600 Workstations addiert. Diese Rechner sind leistungsstärker und können alle Prozesselemente bearbeiten. Somit ergibt sich eine Rechneranzahl von $n_s = 25$ und ein Effizienzwert von $E_p = 0,7$. Wird nur der Zeitraum 16:00 – 20:00 am 20.07.2016 betrachtet, in dem die meisten Evaluationen durchgeführt wurden (334), ergibt sich ein Speedup von $S_p = 20,9$ und eine Effizienz von $E_p = 0,84$.

Die Werte von Speedup und Effizienz stellen sehr gute Ergebnisse dar und zeigen, dass das entwickelte Framework selbstständig die zur Verfügung stehenden Rechner des Clusters verwaltet und diese flexibel zur Bearbeitung von Evaluationsprozessen einsetzt. Während einer laufenden Optimierung können andere Anwender jederzeit interaktiv an den Rechnern des Clusters arbeiten. Dies kann spontan oder periodisch erfolgen. Sobald ein Anwender beginnt, einen Rechner interaktiv zu nutzen, wird die Bearbeitung des Prozesselementes pausiert bzw. nach längerem Pausieren beendet. Das Prozesselement wird anschließend einer freien Ressource zugewiesen. Wird der interaktiv genutzte Rechner wieder freigegeben, steht er automatisch für die Bearbeitung neuer Prozesselemente zur Verfügung. Die laufende Optimierung benötigt bei dieser Art von Störungen zwangsläufig mehr Zeit, bleibt jedoch trotzdem effizient, da die vorhanden Ressourcen durch dynamische Allokation flexibel neu verteilt werden und Leerlauf stets automatisch minimiert wird.

5.6 Handlungsempfehlungen zur effizienten Durchführung einer Optimierung

Aus den Erkenntnissen der beschriebenen Fallstudien werden Handlungsempfehlungen zur effizienten Durchführung einer Optimierung definiert. Eine robuste Parametrisierung des verwendeten Evaluationsmodells (s. Abschnitt 2.5.1) sowie die Auswahl einer geeigneten Optimierungsstrategie für die Bearbeitung der Optimierungsaufgabe (s. Abschnitt 2.8)

werden hierbei vorausgesetzt. Folgende Empfehlungen sollten im Vorfeld einer Optimierung berücksichtigt werden:

- **Wahl der Parallelisierungsmethode:** Die parallele Bearbeitung des Optimierungsproblems sollte in jedem Fall angestrebt werden. Bei sehr kurzen Laufzeiten der Evaluationsprozesse können die Overhead-Zeiten der parallelen Bearbeitung länger sein als die eigentliche Evaluation. In diesem Fall kann es effizienter sein, die Optimierung durchgängig rein sequentiell durchzuführen, da hierbei kein Overhead entsteht [SG15]. Liegt die Laufzeit des Evaluationsprozesses deutlich unterhalb der gemittelten Overhead-Zeit von 10 s (s. Anhang A.6.1), sollte der Evaluationsprozess sequentiell durchgeführt werden.

Auch die sequentielle Bearbeitung einzelner Prozesselemente nach dem Simultaneous Optimization-Ansatz sollte geprüft werden. Insbesondere wenn nur wenige Ressourcen zur Verfügung stehen und einige Prozesselemente numerisch nicht sehr aufwendig sind bzw. die Laufzeit im Bereich der Overhead-Zeit liegt, kann durch die sequentielle Bearbeitung dieser Prozesselemente (z. B. am Anfang bzw. am Ende des DAG) auf dem Master-Rechner die Gesamtlaufzeit reduziert werden. Als Entscheidungshilfe dienen die idealisierten Laufzeiten der Parallelisierungsmethoden und der Grenzwert der Simultaneous Optimization (s. Gleichungen (5.1) – (5.3)).

- **Reduzierung der Laufzeit des Evaluationsprozesses:** Es sollte stets versucht werden, die Laufzeit des Evaluationsprozesses möglichst gering zu halten. Dies kann erreicht werden, indem die innerhalb des Evaluationsmodells verwendeten Modelle bei ausreichender Ergebnisqualität möglichst einfach gehalten werden [SG15]. Bei der Verwendung von FE-Simulationen sollte die Größe des FE-Modells an den zur Verfügung stehenden Arbeitsspeicher der Ressourcen angepasst werden bzw. Anforderungen an die Hardware der Ressourcen formuliert werden (s. Abschnitt 5.3.6).

Führen diese Maßnahmen nicht zur gewünschten Laufzeitreduzierung der Optimierung sollte erst dann das Optimierungsmodell bzw. der Optimierungsalgorithmus angepasst werden [SG15].

- **Dekomposition des Evaluationsmodells:** Die Dekomposition des Evaluationsmodells erfolgt durch den Anwender mit der Definition der Prozesselemente des Evaluationsprozesses als DAG. Die Prozesselemente sollten gemäß der verwendeten Ressourcen definiert werden, z. B. gemäß der benötigten Software oder Hardware.
- **Reduzierung der Kommunikationszeit:** Bei der Dekomposition des Evaluationsmodells sollte eine möglichst geringe Kommunikationszeit durch die Übertragung geringer Datenmengen angestrebt werden, um somit die Overhead-Zeit gering zu halten. Dies wird erreicht, indem die Auswertung von Ergebnisdateien einer Simulation direkt im Anschluss auf dem Rechner erfolgt, auf dem die Simulation durchgeführt wurde. Danach können die in der Regel sehr großen Ergebnisdateien gelöscht und nur die Datei mit den Werten der relevanten Zustandsvariablen übertragen werden.
- **Größe der Parallelisierungsdomäne:** Ist die Größe der Parallelisierungsdomäne frei wählbar und führt eine größere Parallelisierungsdomäne zu besseren Ergebnissen (z. B. bei der Anzahl der Stützstellen eines Latin-Hypercube-Versuchsplans oder der Populationsgröße eines GA), sollte die Parallelisierungsdomäne immer größer als die Anzahl der Ressourcen gewählt werden, da hierbei größere Effizienzwerte erzielt werden können (s. Abschnitt 5.5.2). Dabei wird vorausgesetzt, dass die Anzahl der Ressourcen ebenfalls bekannt ist.

6 Zusammenfassung und Ausblick

In der vorliegenden Arbeit wird die Forschungsfrage der effizienten Bearbeitung von Optimierungsaufgaben in der Produktentwicklung durch den Einsatz von Parallelisierungsmethoden unter Berücksichtigung der verfügbaren Ressourcen innerhalb einer Organisation beantwortet.

Hierzu werden zunächst die technischen Grundlagen der Produktoptimierung erarbeitet, da diese essentiell für die Durchführung einer effektiven und effizienten Optimierung von Produkten sind. Basierend auf der Analyse verschiedener Klassifizierungen von Optimierungsmethoden und -verfahren aus der Literatur wird eine eigene Darstellung entwickelt, welche die vollständige Einordnung der in der Produktentwicklung etablierten Optimierungsmethoden, -verfahren und -algorithmen ermöglicht und gleichermaßen ein prinzipielles Vorgehensmodell für die Festlegung von Optimierungsstrategien zur Bearbeitung von Optimierungsaufgaben darstellt. Multikriterielle und multidisziplinäre Optimierungsstrategien nehmen in der Produktentwicklung aufgrund der unterschiedlichen und oft gegenläufigen Anforderungen an ein Produkt eine besondere Rolle ein. Zudem wird herausgestellt, wie durch die gezielte Kombination von Optimierungsmethoden und -verfahren die Effektivität und die Effizienz von Optimierungen gesteigert werden können.

Weiterhin wird die Relevanz von Analogiebetrachtungen für die Entwicklung von Optimierungsverfahren und Produktentwicklungsmethoden betrachtet. Von besonderer Bedeutung sind hierbei Analogien zur biologischen Evolution. Diese bilden die Grundlage evolutionärer Algorithmen, welche eine hohe Wahrscheinlichkeit besitzen, das globale Optimum eines Problems zu finden und sich wie verschiedene andere stochastische Optimierungsverfahren aufgrund ihrer Funktionsweise zur Parallelisierung eignen. Weiterhin erfolgt eine Beschreibung des aktuellen Forschungsstands der Autogenetischen Konstruktionstheorie, in der die gesamte Produktentwicklung als evolutionärer Prozess betrachtet wird. Dabei wird herausgestellt, wie insbesondere bei komplexen Evaluationsprozessen durch die entwickelte Parallelisierungsmethode auch hier die Effizienz in der Anwendung erhöht werden kann.

Die Aktivitäten und Prozesse in der Produktentwicklung können auch als kontinuierlicher Optimierungsprozess angesehen werden, wodurch zwischen diesen beiden eine gewisse Konformität besteht, die weitere Analogiebetrachtungen zulässt. So stellen Analogien zum Verhalten interdisziplinärer Entwicklungsteams bei der Produktentwicklung bereits die Grundlage einiger stochastischer Optimierungsverfahren dar. Auf Basis dieser Analogiebetrachtung erfolgt auch die Entwicklung einer Methode zur effizienten Parallelisierung von Evaluationsprozessen von Optimierungen. Als Vorbild dienen hierbei ebenfalls interdisziplinäre Entwicklungsteams, in denen die einzelnen Teammitglieder unterschiedliche Qualifikationen besitzen und je nach Qualifikation unterschiedliche Aufgaben bearbeiten können. Das Ziel bei der Organisation und Aufgabenverteilung ist es, die gemeinsame Gesamtaufgabe möglichst effizient zu bearbeiten und Leerlauf zu minimieren. Zudem werden innerhalb eines funktionierenden Teams selbstständig Ausfälle von Teammitgliedern kompensiert sowie zusätzliche Teammitglieder integriert.

Zur theoretischen Umsetzung dieses Ansatzes werden unter den Begriffen Concurrent Optimization und Simultaneous Optimization die Methoden des Concurrent Engineering und des Simultaneous Engineering, welche bereits erfolgreich zur Parallelisierung von

Produktentwicklungsprozessen genutzt werden, auf die Anwendung in Optimierungsverfahren übertragen. Diese Form der Parallelisierung basiert nur auf der Verfügbarkeit und der Stabilität der für einen Evaluationsprozess benötigten Informationen und Ressourcen, wodurch eine hohe Flexibilität in der Anwendung sowie die dynamische Reaktion auf spontane Veränderungen der Ressourcen ermöglicht werden. Da bei der Evaluation von Produktvarianten in einer Optimierung in der Regel kommerzielle Software verwendet wird, deren Verfügbarkeit den Grad der Parallelisierung und somit die Effizienz der Optimierung einschränken kann, wird die konventionelle Ressourcendefinition, welche primär die zur Verfügung stehende Hardware beinhaltet, um die vorhandene Software und die zur Verfügung stehenden Softwarelizenzen erweitert. Die Methode unterstützt die Dekomposition des Optimierungsmodells sowie des Evaluationsmodells und kann somit bei mono- und multidisziplinären Optimierungen in Verbindung mit verschiedenen Optimierungsalgorithmen verwendet werden.

Die prototypische Umsetzung dieser Methode erfolgt durch die Entwicklung eines Frameworks, in dem die Evaluationsprozesse von Optimierungen effizient parallelisiert werden. Dieses beinhaltet die Integration eines Workstation-Clusters unter Verwendung des Cluster-Management-Systems HTCondor, wodurch die freien Rechenkapazitäten vorhandener Workstations innerhalb der Organisation genutzt werden können und somit eine sehr kosteneffiziente verteilte IT-Umgebung bereitgestellt wird. Weiterhin kann innerhalb des Frameworks der lokale Master-Rechner, auf dem die Optimierung nach dem Master-Worker-Schema durchgeführt wird, als zusätzliche Ressource verwendet werden. Dies ermöglicht die Bearbeitung von Prozesselementen, welche eine grafische Benutzeroberfläche bei der Ausführung erfordern und somit nicht in einer Cluster-Umgebung ausgeführt werden können. Die Bearbeitung dieser Prozesselemente erfolgt auf dem lokalen Master-Rechner der Optimierung nach dem Simultaneous Optimization-Ansatz. Zur Bereitstellung evolutionärer und genetischer Algorithmen wird das Optimierungssystem NOA in das Framework integriert. Die Schnittstelle zum Anwender bildet das Optimierungssystem DAG2OPT. Bis auf das Erstellen der Skripte zur Automatisierung der Evaluationsprozesse können alle Aktivitäten zur Durchführung einer Optimierung über eine grafische Benutzeroberfläche durchgeführt werden. DAG2OPT dient somit als Front-End für den Anwender zur Konfiguration und Verwendung des gesamten Frameworks, zur Definition des Optimierungs- und Evaluationsmodells und stellt zudem verschiedene Algorithmen zur Durchführung von Optimierungen oder DoE-Studien bereit.

Weiterhin erfolgt in DAG2OPT die Optimierung der Prioritäten der Prozesselemente des Evaluationsprozesses, durch die eine weitere Effizienzsteigerung des Evaluationsprozesses und somit der gesamten Optimierung möglich ist. Im Rahmen der Implementierung werden anhand von drei Testszenarien unterschiedlicher Komplexität verschiedene Scheduling-Verfahren gemäß ihrer Eignung zur Ermittlung der optimalen Prozessprioritäten analysiert. Aus den durchgeführten Untersuchungen geht hervor, dass die optimalen Prioritäten nicht durchgängig mit den etablierten Heuristiken ermittelt werden können, da die Größe der Parallelisierungsdomäne einen entscheidenden Einfluss auf die optimalen Prioritätswerte der Prozesselemente hat und keine der Heuristiken allgemeingültig ist. Die Verwendung von Optimierungsverfahren stellt im Vergleich zu Heuristiken die wesentlich effektivere und flexiblere Lösung zur Ermittlung der optimalen Prioritäten dar. Der genetische Algorithmus NSGA-II generiert in den durchgeführten Untersuchungen stets die optimalen Prioritätswerte und wird daher in DAG2OPT implementiert, um im Vorfeld einer Produktoptimierung die optimalen Prioritätswerte der Prozesselemente des Evaluationsprozesses gemäß der erforderten und der zur Verfügung stehenden Ressourcen

zu ermitteln. Untersuchungen zur Laufzeit der Allokationssimulation, welche der Prioritätsoptimierung zugrunde liegt, zeigen, dass die Optimierung der Prozessprioritäten den generellen Arbeitsprozess nicht einschränkt. Der Anwender wird hierbei über die prognostizierte Laufzeit informiert. In den durchgeführten Untersuchungen konnte durch die Anwendung der optimierten Prioritäten die Laufzeit des Evaluationsprozesses einer Produktoptimierung um maximal 10,4 % verkürzt werden. Die mögliche Laufzeitverkürzung hängt dabei stark von der Laufzeit der Prozesselemente ab und ist nicht als allgemeingültig zu betrachten.

Die Validierung des Frameworks erfolgt anhand von drei Fallstudien, welche charakteristische multidisziplinäre Optimierungsaufgaben aus realen Entwicklungsprojekten darstellen. Dabei liegt besonderer Fokus auf dem flexiblen Einsatz des Optimierungssystems und der dynamischen Allokation der zur Verfügung stehenden Ressourcen. Die Durchführung der Fallstudien erfolgt in einem heterogenen Workstation-Cluster eines universitären Computer-Labors. Die IT-Umgebung ist daher vergleichbar mit der IT-Umgebung kleiner und mittlerer Unternehmen.

Die im Rahmen der prototypischen Umsetzung der Parallelisierungsmethode gestellten Anforderungen werden durch das Framework vollständig erfüllt. Die Basis für die effiziente Parallelisierung der Prozesselemente des Evaluationsprozesses bildet die vollständige Modelldekomposition des Optimierungs- und des Evaluationsmodells. Bei der parallelen Bearbeitung werden somit die spezifischen Anforderungen jedes Prozesselementes und die zur Verfügung stehenden Ressourcen berücksichtigt und die optimale Konfiguration zur schnellstmöglichen Durchführung einer Optimierung erstellt. Gemäß der Einstellungen und Anforderungen an die Prozesselemente werden automatisch die leistungsfähigsten Workstations für die Bearbeitung ausgewählt. Zudem erfolgt die Auswahl der Parallelisierung für jedes Prozesselement individuell. Die Prozesselemente können dabei in dem Workstation-Cluster oder lokal auf dem Master-Rechner der Optimierung bearbeitet werden. Dies erlaubt z. B. die Integration von Prozesselementen, welche eine grafische Oberfläche benötigen und nicht in einer Cluster-Umgebung ausgeführt werden können. Die in das Framework implementierte dynamische Allokation der Prozesselemente ermöglicht die flexible Skalierbarkeit des Clusters. Zusätzlich verfügbare Ressourcen können flexibel in den Cluster integriert werden. Ausfälle von Ressourcen werden durch den Cluster kompensiert. Während laufender Optimierungen ist es somit jederzeit möglich, spontan oder periodisch interaktiv an den Workstations des Clusters zu arbeiten, was in den Ergebnissen der Fallstudien erkennbar ist. Aus den Erkenntnissen der Fallstudien werden Handlungsempfehlungen zur effizienten Durchführung von Optimierungen in dem entwickelten Framework formuliert.

Die durchgeführten Fallstudien spiegeln ein industrienahes Umfeld wider, in dem freie Workstations als Ressourcen in einem Cluster zur Verfügung stehen können, z. B. während Pausen, in der Nacht oder am Wochenende. Das entwickelte Framework kann sowohl in homogenen als auch in heterogenen Workstation-Clustern eingesetzt werden. Der Workstation-Cluster kann dabei kostenneutral aus einer beliebigen Anzahl in einem Netzwerk verbundener Workstations realisiert werden. Zusätzliche Hardware wird nicht benötigt. Zudem besteht keine Notwendigkeit eines gemeinsamen Speichers. Der Datentransfer erfolgt direkt zwischen den Workstations des Clusters. Jede zur Verfügung stehende Workstation kann dem Cluster schnell und einfach hinzugefügt oder aus dem Cluster entfernt werden, auch während laufenden Optimierungen. Die Workstations des Clusters stehen daher jederzeit zur interaktiven Arbeit zur Verfügung. Die interaktive Arbeit wird nicht

beeinträchtigt. Aufgrund der kostenneutralen Implementierung eignet sich das Framework daher auch für die Verwendung in kleinen und mittleren Unternehmen, in denen aufgrund der hohen Anschaffungskosten kein oder nur eingeschränkter Zugang zu HPC-Supercomputern besteht.

Da die Workstations des Clusters aufgrund der interaktiven Arbeit standardmäßig über Grafikkarten verfügen, können auch die GPU der Workstations zur Bearbeitung dazu geeigneter Prozesselemente genutzt werden (s. Abschnitt 4.4.1), was einen zusätzlichen Vorteil des Frameworks darstellt. GPU stehen in kommerziellen Supercomputern in der Regel nicht bzw. nur bei kostenintensiver Konfiguration zur Verfügung. Wird bei der Bearbeitung der Prozesselemente kommerzielle Software verwendet, z. B. CAE-Systeme, können konventionelle Arbeitsplatzlizenzen genutzt werden. Spezielle und oft kostenintensive Lizenzen zur Verwendung der Software auf HPC-Systemen werden nicht benötigt. Aufgrund der beschriebenen Eigenschaften hebt sich das in dieser Arbeit entwickelte Framework somit deutlich von konventionellen Systemen zur parallelen Bearbeitung von Evaluationsprozessen bei Optimierungen ab.

Ein weiterer Vorteil der effizienten Parallelisierung ergibt sich aus der indirekten Beeinflussung der Effektivität der Optimierungen durch die Anzahl an Evaluationen. War die Laufzeit einer Optimierung bisher z. B. aufgrund von Projektterminen limitiert, was den Einsatz weniger effektiver Optimierungsverfahren bedingte, z. B. deterministische Verfahren, können unter Berücksichtigung der zur Verfügung stehenden Ressourcen nun effektivere Optimierungsverfahren eingesetzt werden, z. B. genetische Algorithmen, welche in der Regel eine größere Anzahl an Evaluationen benötigen. Zudem kann die Effektivität eines genutzten Verfahrens auch über die Größe der Parallelisierungsdomäne beeinflusst werden. So kann z. B. bei genetischen Algorithmen die Populationsgröße erhöht werden, was in der Regel in einer größeren Wahrscheinlichkeit resultiert, das globale Optimum eines Problems zu finden. Auch bei DoE-Studien führt eine größere Anzahl an Stützstellen zu einer genaueren Approximation des Lösungsraumes durch Metamodelle. Eine Empfehlung zur Wahl der Größe der Parallelisierungsdomäne wird in den Handlungsempfehlungen zur effizienten Durchführung von Optimierungen gegeben.

Die Ergebnisse der Fallstudien zeigen auch die Nachteile des entwickelten Frameworks auf. So führten Ausfälle einzelner Ressourcen und des kompletten Clusters zu einer Unterbrechung der gesamten Optimierung, obwohl die Ressourcen nach dem Ausfall wieder zur Verfügung standen. Die Optimierung musste anschließend manuell gestartet werden. Im Zeitraum der Unterbrechung entstand somit unnötiger Leerlauf der Ressourcen. Obwohl die Ursachen dieser Ausfälle behoben wurden und anschließend ein stabiler Einsatz gewährleistet werden konnte, zeigt dies prinzipiell eine mögliche Fehleranfälligkeit des Frameworks. Eine Form der Abhilfe stellt die Implementierung eines Fehler-Management-Systems dar, welches auf Basis zu langer Wartezeiten von Prozesselementen im Vergleich zur Laufzeit ähnlicher Prozesselemente einen Fehler detektiert und entsprechende Maßnahmen einleitet. Dies kann z. B. die automatische Benachrichtigung des Anwenders per E-Mail oder der Abbruch der Bearbeitung des Prozesselementes sein. Eine weitere Form des Fehler-Managements bildet die automatische Kompensierung von Fehlern beim Abbruch von Prozesselementen. Durch im Hintergrund generierte Metamodelle können die benötigten Zustandsvariablen approximiert und der Optimierung bereitgestellt werden [RB13].

In dem entwickelten Framework erfolgt die Ermittlung der Prioritätswerte im Vorfeld einer Optimierung einmalig durch den Anwender. Dieser Prozess bietet Potential für die Weiterentwicklung des Frameworks. So könnten im Verlauf einer Optimierung die Prioritätswerte der Prozesselemente selbstständig in regelmäßigen Abständen durch das System überprüft und ggf. überschrieben werden. Dies könnte bei einem genetischen Algorithmus z. B. in jeder Generation erfolgen. Aber auch eine permanente Überprüfung durch einen im Hintergrund agierenden Software-Agenten ist denkbar. Diese in definierten Grenzen unabhängig agierende Komponente des Systems stellt zudem eine Möglichkeit dar, die Vererbung der Prioritätswerte durch HTCondor zu umgehen, welche die Fertigstellung eines bereits begonnenen DAG sicherstellen (s. Abschnitt 5.3.4). Der Agent könnte dabei permanent im Hintergrund versuchen, die Prioritäten der ausstehenden Prozesselemente weiter zu optimieren. Dabei könnte jeder Berechnungsjob in der Warteschlange individuell und unabhängig von dessen DAG betrachtet werden. Dieser Ansatz bietet Potential zur weiteren Steigerung der Effizienz von Optimierungen und sollte in der Zukunft näher untersucht werden. Eine weitere Effizienzsteigerung ist zudem durch eine Reduzierung des Overheads möglich. Potential hierzu wird bei der Übertragung der Prozesselemente an den Cluster durch DAG2OPT oder bei der Allokation der Ressourcen durch HTCondor gesehen.

Weiteres Potential bietet die Implementierung zusätzlicher Optimierungsverfahren. Findet eine Dekomposition des Simulationsmodells statt, kann das entwickelte Framework auch zur parallelen Bearbeitung der Evaluationsprozesse deterministischer Optimierungsverfahren genutzt werden. Auch die Implementierung weiterer stochastischer Verfahren wird angestrebt, z. B. Particle Swarm und Simulated Annealing. Zusätzlich sollte die in [GH02] vorgestellte Multi-Algorithmus-Strategie näher untersucht und auf die Verwendung in dem entwickelten Framework hin geprüft werden. Dabei konkurrieren verschiedene Optimierungsalgorithmen parallel miteinander, wodurch die Wahrscheinlichkeit erhöht wird, das globale Optimum eines Problems zu finden.

Dieser Ansatz zeigt zudem weiteres Potential in der Erforschung der Anwendung von Optimierungsverfahren in der Produktentwicklung auf. Analog zur in [KWW15] vorgestellten Auswertung von Simulations- und Messdaten ist eine Auswertung der Optimierungsdaten bezogen auf die jeweilige Optimierungsaufgabe denkbar, um somit den Anwender bei der Konfiguration und Modellerstellung mit Erfahrungswissen zu unterstützen. Von besonderem Interesse wären hierbei der gewählte Optimierungsalgorithmus, die Designvariablen und Zielkriterien, das Evaluationsmodell, die Anzahl durchgeführter Evaluationen sowie die Auswahl einer Lösung aus der Pareto-optimalen Menge der Lösungen.

Auch die Integration weiterer Ressourcen stellt eine Möglichkeit dar, das im Rahmen dieser Arbeit entwickelte Framework weiterzuentwickeln, z. B. durch die Entwicklung von Schnittstellen zu den in der Einleitung erwähnten HPC-Supercomputern und HPC-Cloud-Services. Das Framework wird in der Anwendung nicht als eine reine Alternative zu HPC-Supercomputern und HPC-Cloud-Services, sondern als Ergänzung hierzu gesehen, um somit die mit den steigenden Anforderungen an die Produktentwicklung verbundenen Berechnungs- und Optimierungsaufgaben flexibler und robuster und somit effizienter zu verteilen.

Literaturverzeichnis

- [ACD⁺12] AKBARIYEH, A.; CARRIGAN, T. J.; DENNIS, B. H.; CHAN, W. S.; WANG, B. P.; LAWRENCE, K. L.: *Application of GPU-Based Computing to Large Scale Finite Element Analysis of Three-Dimensional Structures*. In: TOPPING, B. (Hrsg.): *Proceedings of The Eighth International Conference on Engineering Computational Technology*. Dubrovnik, Kroatien, 04.–07.09.2012. Stirlingshire, UK: Civil-Comp Press, DOI 10.4203/ccp.100.6
- [ACTR02] ANSOLA, R.; CANALES, J.; TÁRRAGO, J. A.; RASMUSSEN, J.: *An integrated approach for shape and topology optimization of shell structures*. In: *Computers & Structures* 80 (2002), Nr. 5-6, S. 449–458. ISSN 0045–7949. DOI 10.1016/S0045-7949(02)00019-6
- [ACTR04] ANSOLA, R.; CANALES, J.; TARRAGO, J. A.; RASMUSSEN, J.: *Combined shape and reinforcement layout optimization of shell structures*. In: *Structural and Multidisciplinary Optimization* 27 (2004), Nr. 4, S. 219–227. ISSN 1615–147X. DOI 10.1007/s00158-004-0399-7
- [AD11] APPELYARD, J.; DRIKAKIS, D.: *Higher-order CFD and interface tracking methods on highly-Parallel MPI and GPU systems*. In: *Computers & Fluids* 46 (2011), Nr. 1, S. 101–105. ISSN 0045–7930. DOI 10.1016/j.compfluid.2010.10.019
- [AEH⁺10] ACHENBACH, M.; EDLER, J.; HELLMIG, R. J.; MATTHECK C.; MOLDENHAUER, H.; SACHS, W.; TESARI, I.: *Entwicklung von effizienten, einfach anzuwendenden Konstruktionsprinzipien für technische Bauteile nach dem Vorbild der Natur*. Schlussbericht zum BMBF-geförderten Verbundprojekt, Förderkennzeichen 01 RI 0638, 2010
- [AHC15] ANDREASEN, M. M.; HANSEN, C. T.; CASH, P.: *Conceptual Design: Interpretations, mindset and models*. 2nd ed. Cham: Springer International Publishing, 2015. ISBN 978–3–319–19838–5
- [ALG12] ALGOLIC: *QNodesEditor - Qt nodes/ports-based data processing flow editor*. <http://alcoholic.eu/qnodeseditor-qt-nodesports-based-data-processing-flow-editor/>, Abruf: 04.06.2015
- [Alt12] ALT, H. W.: *Lineare Funktionalanalysis: Eine anwendungsorientierte Einführung*. 6. Aufl. Berlin, Heidelberg: Springer-Verlag, 2012. ISBN 978–3–642–22260–3
- [Alt14] ALTAIR ENGINEERING: *OptiStruct 13.0 User's Guide*. Altair Engineering Inc., 2014
- [Alt15] ALTAIR ENGINEERING: *Practical Aspects of Structural Optimization: A Study Guide*. 2nd Edition. Altair Engineering, Inc., 2015

- [AMY15] ABI AKLE, A.; MINEL, S.; YANNOU, B.: *Graphical Support Adapted to Designers for the Selection of an Optimal Solution in Design by Shopping*. In: WEBER, C.; HUSUNG, S.; CASCINI, G.; CANTAMESSA, M.; MARJANOVIĆ, D. (Hrsg.): *Proceedings of the 20th International Conference on Engineering Design (ICED 15), Vol. 6: Design Methods and Tools - Part 2*. Mailand, Italien, 27.–30.07.2015, S. 215–224. ISBN 978–1–90467–069–8
- [AT02] ALBA, E.; TOMASSINI, M.: *Parallelism and Evolutionary Algorithms*. In: *IEEE Transactions on Evolutionary Computation* 6 (2002), Nr. 5, S. 443–462. DOI 10.1109/TEVC.2002.800880
- [AW05] ANDERSON, M. J.; WHITCOMB, P. J.: *RSM Simplified: Optimizing Processes Using Response Surface Methods for Design of Experiments*. New York, N.Y.: Productivity Press, 2005. ISBN 978–1–56327–297–4
- [Bak85] BAKER, J. E.: *Adaptive Selection Methods for Genetic Algorithms*. In: GREFENSTETTE, J. J. (Hrsg.): *Proceedings of the First International Conference on Genetic Algorithms and Their Applications*. Pittsburg, PA, 24.–26.07.1985. Mahwah, NJ: Lawrence Erlbaum Associates, S. 101–111. ISBN 0–8058–0426–9
- [Bak87] BAKER, J. E.: *Reducing Bias and Inefficiency in the Selection Algorithm*. In: GREFENSTETTE, J. J. (Hrsg.): *Proceedings of the Second International Conference on Genetic Algorithms and Their Applications*. Cambridge, MA, 28.–31.07.1987. Hillsdale, NJ: Erlbaum, ISBN 0–8058–0158–8
- [BB60] BOX, G. E. P.; BEHNKEN, D. W.: *Some New Three Level Designs for the Study of Quantitative Variables*. In: *Technometrics* 2 (1960), Nr. 4, S. 455–475. ISSN 0040–1706. DOI 10.1080/00401706.1960.10489912
- [BBKS15] BENGEL, G.; BAUN, C.; KUNZE, M.; STUCKY, K.-U.: *Masterkurs Parallele und Verteilte Systeme: Grundlagen und Programmierung von Multicoreprozessoren, Multiprozessoren, Cluster und Grid*. 2., Aufl. Wiesbaden: Springer Fachmedien, 2015. ISBN 978–3–83481–671–9
- [BC11] BELEGUNDU, A. D.; CHANDRUPATLA, T. R.: *Optimization Concepts and Applications in Engineering*. 2nd ed. New York: Cambridge University Press, 2011. ISBN 978–0–521–87846–3
- [Bes06] BESTLE, D.: *Strategien der Mehrkriterien-Optimierung*. In: GRABE, J. (Hrsg.): *Optimierung in der Geotechnik*. Hamburg, 10.10.2006, Technische Universität Hamburg-Harburg, Institut für Geotechnik und Baubetrieb, S. 45–57. ISBN 3–93631–013–0
- [BFM97] BÄCK, T.; FOGEL, D. B.; MICHALEWICZ, Z. (Hrsg.): *Handbook of Evolutionary Computation*. Bristol, Philadelphia, New York: Institute of Physics Pub., Oxford University Press, 1997. ISBN 978–0–75030–895–3
- [BHK11] BELAY, A. M.; HELO, P.; KASIE, F. M.: *Concurrent Engineering Yesterday, Today and Tomorrow*. In: FREY, D. D.; FUKUDA, S.; ROCK, G. (Hrsg.): *Improving Complex Systems Today*. London: Springer-Verlag, 2011, S. 425–432. ISBN 978–0–85729–798–3. DOI 10.1007/978-0-85729-799-0_50

- [BHSS92] BLOEBAUM, C. L.; HAJELA, P.; SOBIESZCZANSKI-SOBIESKI, J.: *Non-Hierarchical System Decomposition in Structural Optimization*. In: *Engineering Optimization* 19 (1992), Nr. 3, S. 171–186. ISSN 0305–215X. DOI 10.1080/03052159208941227
- [Bis89] BISHOP, J. M.: *Stochastic Searching Networks*. In: *First IEE International Conference on Artificial Networks*. 16.–18.10.1989, Institution of Electrical Engineers (IEE conference publications), S. 329–331. ISBN 978–0–85296–388–3
- [BK97] BRAUN, R. D.; KROO, I. M.: *Development and Application of the Collaborative Optimization Architecture in a Multidisciplinary Design Environment*. In: ALEXANDROV, N. M.; HUSSAINI, M. Y. (Hrsg.): *Multidisciplinary Design Optimization: Proceedings of the ICASE/NASA Langley Workshop on Multidisciplinary Design Optimization*. Hampton, Virginia, USA, 13.–16.03.1995. Philadelphia: Society for Industrial and Applied Mathematics, 1997, S. 98–117. ISBN 0–89871–359–5
- [BM06] BAUKE, H.; MERTENS, S.: *Cluster Computing: Praktische Einführung in das Hochleistungsrechnen auf Linux-Clustern*. Berlin, Heidelberg, New York: Springer-Verlag, 2006. ISBN 978–3–540–42299–0
- [BMB12] BMBF (Hrsg.): *Innovative Produkte effizient entwickeln „Forschung für die Produktion von morgen“: Projektportraits*. Bonn, Berlin: Bundesministerium für Bildung und Forschung, Referat Forschung für Produktion, Dienstleistung und Arbeit, 2012
- [Boo87] BOOKER, L.: *Improving Search in Genetic Algorithms*. In: DAVIS, L. (Hrsg.): *Genetic Algorithms and Simulated Annealing*. London, Los Altos, CA: Pitman and Morgan Kaufmann Publishers, 1987, S. 61–73. ISBN 978–0–93461–344–6
- [Bou15] BOUWMAN, W.: *Diagram Editor*. <http://www.windel.nl/?section=pyqtdiagrameditor>, Abruf: 04.06.2015
- [BPK15] BRANDT-POOK, H.; KOLLMEIER, R.: *Softwareentwicklung kompakt und verständlich: Wie Softwaresysteme entstehen*. 2. Aufl. Wiesbaden: Springer Fachmedien, 2015. ISBN 978–3–658–10875–5
- [Bra96] BRAUN, R. D.: *Collaborative Optimization: An Architecture for Large-Scale Distributed Design*. Dissertation. Stanford University, Stanford, CA, 1996
- [Bro70] BROYDEN, C. G.: *The Convergence of a Class of Double-rank Minimization Algorithms 1. General Considerations*. In: *IMA Journal of Applied Mathematics* 6 (1970), Nr. 1, S. 76–90. DOI 10.1093/imamat/6.1.76
- [BS04] BENDSØE, M. P.; SIGMUND, O.: *Topology optimization: Theory, methods, and applications*. 2nd ed., corrected print. Berlin, New York: Springer-Verlag, 2004. ISBN 978–3–64207–698–5
- [BSG⁺13] BRONSTEIN, I. N.; SEMENDJAEW, K. A.; GROSCHE, G.; ZIEGLER, V.; ZIEGLER, D.: *Springer-Taschenbuch der Mathematik*. 3. Aufl. Wiesbaden: Springer Spektrum, 2013. ISBN 978–3–8351–0123–4

- [BSS94] BAIER, H.; SEESSELBERG, C.; SPECHT, B.: *Optimierung in der Strukturmechanik*. Braunschweig, Wiesbaden: Vieweg, 1994. ISBN 3-52808-899-0
- [BSS06] BAZARAA, M. S.; SHERALI, H. D.; SHETTY, C. M.: *Nonlinear Programming: Theory and Algorithms*. 3. Auflage. Hoboken, NJ: Wiley-Interscience, 2006. ISBN 9-780-47148-600-8
- [BT76] BELL, T. E.; THAYER, T. A.: *Software Requirements: Are They Really a Problem?* In: *Proceedings of the 2nd International Conference on Software Engineering*. Los Alamitos: IEEE Computer Society Press, S. 61-68
- [BT95] BLICKLE, T.; THIELE, L.: *A Comparison of Selection Schemes used in Genetic Algorithms*. TIK-Report, Nr. 11, Version 2, 1995
- [BV94a] BERCSEY, T.; VAJNA, S.: *Ein Autogenetischer Ansatz für die Konstruktions-theorie: Beitrag zur vollständigen Beschreibung des Konstruktionsprozesses, Teil 1*. In: *CAD-CAM Report 2* (1994), S. 66-71. ISSN 0930-7117
- [BV94b] BERCSEY, T.; VAJNA, S.: *Ein Autogenetischer Ansatz für die Konstruktions-theorie: Beitrag zur vollständigen Beschreibung des Konstruktionsprozesses, Teil 2*. In: *CAD-CAM Report 3* (1994), S. 98-105. ISSN 0930-7117
- [Cal15] CALLIGRA: *Calligra Flow*. <https://www.calligra.org/flow/>, Abruf: 04.06.2015
- [Cam12] CAMPAGNOLA, L.: *PyQtGraph: Scientific Graphics and GUI Library for Python*. <http://www.pyqtgraph.org/>, Abruf: 04.06.2015
- [Cav13] CAVAZZUTI, M.: *Optimization Methods: From Theory to Design: Scientific and Technological Aspects in Mechanics*. Berlin, New York: Springer-Verlag, 2013. ISBN 978-3-64231-186-4
- [CB11] CZAPIŃSKI, M.; BARNES, S.: *Tabu Search with two approaches to parallel flowshop evaluation on CUDA platform*. In: *Journal of Parallel and Distributed Computing* 71 (2011), Nr. 6, S. 802-811. ISSN 0743-7315. DOI 10.1016/j.jpdc.2011.02.006
- [CCK99] CAMPBELL, M. I.; CAGAN, J.; KOTOVSKY, K.: *A-Design: An Agent-Based Approach to Conceptual Design in a Dynamic Environment*. In: *Research in Engineering Design* 11 (1999), Nr. 3, S. 172-192. ISSN 0934-9839. DOI 10.1007/s001630050013
- [CDM92] COLORNI, A.; DORIGO, M.; MANIEZZO, V.: *Distributed Optimization by Ant Colonies*. In: VARELA, F. J.; BOURGINE, P. (Hrsg.): *Toward a Practice of Autonomous Systems*. Paris, Frankreich, 11.-13.12.1991. Cambridge, MA: MIT Press, S. 134-142. ISBN 978-0-26272-019-9
- [Cen15] CENTER FOR HIGH THROUGHPUT COMPUTING: *HTCondorTM Version 8.4.3 Manual*. Center for High Throughput Computing, University of Wisconsin-Madison, 2015
- [CES89] CARUANA, R. A.; ESHELMAN, L. J.; SCHAFFER, J. D.: *Representation and Hidden Bias II: Eliminating Defining Length Bias in Genetic Search via Shuffle Crossover*. In: *Proceedings of the 11th International Joint Conference on Artificial Intelligence - Volume 1*. San Francisco: Morgan Kaufmann Publishers, S. 750-755

- [CKO00] CORNE, D. W.; KNOWLES, J. D.; OATES, M. J.: *The Pareto Envelope-Based Selection Algorithm for Multiobjective Optimization*. In: GOOS, G.; HARTMANIS, J.; VAN LEEUWEN, J.; SCHOENAUER, M.; DEB, K.; RUDOLPH, G.; YAO, X.; LUTTON, E.; MERELO, J. J.; SCHWEFEL, H.-P. (Hrsg.): *Parallel Problem Solving from Nature PPSN VI*. Berlin, Heidelberg: Springer, 2000, S. 839–848. ISBN 978-3-540-41056-0. DOI 10.1007/3-540-45356-3_82
- [CKR⁺07] COUVARES, P.; KOSAR, T.; ROY, A.; WEBER, J.; WENGER, K.: *Workflow Management in Condor*. In: TAYLOR, I. J.; DEELMAN, E.; GANNON, D.; SHIELDS, M. S. (Hrsg.): *Workflows for e-science*. London: Springer-Verlag, 2007, S. 357–375. ISBN 978-1-84628-519-6
- [Cle92] CLEETUS, K. J.: *Definition of Concurrent Engineering*. CERC Technical Report Series, CERC-TR-RN-92-003, Concurrent Engineering Research Center, West Virginia University, 1992
- [Coh15] COHEN, L.: *JPPF: The open source grid computing solution*. <http://www.jpff.org/>, Abruf: 21.12.2015
- [Cöl16] CÖLLEN, H. C.: *Prioritätenbasierte Optimierung zur Parallelisierung von rechenintensiven Problemen in der Produktentwicklung*. Bachelorarbeit. Otto-von-Guericke-Universität Magdeburg, Magdeburg, 2016
- [Cou08a] COUNCIL ON COMPETITIVENESS: *Reflect. Council on Competitiveness and USC-ISI In-Depth Study of Technical Computing End Users and HPC*. <http://www.compete.org/reports/all/421-reflect>, Abruf: 19.10.2016
- [Cou08b] COUNCIL ON COMPETITIVENESS: *Reveal. Council on Competitiveness and USC-ISI Broad Study of Desktop Technical Computing End Users and HPC*. <http://www.compete.org/reports/all/420-reveal>, Abruf: 19.10.2016
- [Cou14] COUNCIL ON COMPETITIVENESS: *Solve. The Exascale Effect: the Benefits of Supercomputing Investment for U.S. Industry*. <http://www.compete.org/reports/all/2695>, Abruf: 19.10.2016
- [Cou16] COUNTEREXPLOITATION: *LSP-Fix*. <http://cexx.org/lspfix.htm>, Abruf: 10.02.2016
- [CP98] CANTÚ-PAZ, E.: *A Survey of Parallel Genetic Algorithms*. Department of Computer Science and Illinois Genetic Algorithms Laboratory, University of Illinois at Urbana-Champaign. 1998
- [CRG14] CHALLIS, V. J.; ROBERTS, A. P.; GROTHOWSKI, J. F.: *High resolution topology optimization using graphics processing units (GPUs)*. In: *Structural and Multidisciplinary Optimization* 49 (2014), Nr. 2, S. 315–325. ISSN 1615-147X. DOI 10.1007/s00158-013-0980-z
- [CSDV02] CRAIG, K.; STANDER, N.; DOOGE, D.; VARADAPPA, S.: *MDO of Automotive Vehicle for Crashworthiness and NVH Using Response Surface Methods*. In: *A collection of technical papers: 9th AIAA/ISSMO Symposium on Multidisciplinary Analysis and Optimization*. Atlanta, Georgia, USA, 04.–06.09.2002, American Institute of Aeronautics and Astronautics, S. 1930–1940. ISBN 978-1-56347-550-4

- [CSX05] CHEN, Z.; SINGH, V.; XU, J.: *Efficient Job Scheduling Algorithms with Multi-Type Contentions*. In: *Journal of Combinatorial Optimization* 10 (2005), Nr. 2, S. 179–197. DOI 10.1007/s10878-005-2272-z
- [Cur44] CURRY, H. B.: *The method of steepest descent for nonlinear minimization problems*. In: *Quarterly Journal of Mechanics and Applied Mathematics* 2 (1944), S. 258–261. ISSN 0033-5614
- [DA06] DONG, F.; AKL, S. G.: *Scheduling Algorithms for Grid Computing: State of the Art and Open Problems*. Technical Report No. 2006-504, Queen's University, Kingston, Ontario, 2006
- [Dav59] DAVIDON, W. C.: *Variable Metric Method for Minimization*. A.E.C. Research and Development Report, ANL-5990 Rev., 1959
- [DFJ54] DANTZIG, G.; FULKERSON, R.; JOHNSON, S.: *Solution of a Large-Scale Traveling-Salesman Problem*. In: *Journal of the Operations Research Society of America* 2 (1954), Nr. 4, S. 393–410. DOI 10.1287/opre.2.4.393
- [Die14] DIETRICH, T.: *Neue virtuelle Methoden zur Integration einer numerischen Fußgängerschutz-Optimierung in den Entwicklungsprozess einer Motorhaube*. Dissertation. Technische Universität München, München, 2014
- [DMM03] DEB, K.; MOHAN, M.; MISHRA, S.: *A Fast Multi-objective Evolutionary Algorithm for Finding Well-Spread Pareto-Optimal Solutions*. Tech. Report, Kanpur Genetic Algorithms Laboratory (KanGAL), Report Number 2003002, 2003
- [DO99] DUFFY, A. H. B.; O'DONNELL, F. J.: *A Design Research Approach*. In: MORTENSEN, N. H.; SIGURJONSSON, J. (Hrsg.): *Critical enthusiasm*. Trondheim: Norges Teknisk Naturvitenskapelige Universitet, S. 33–40. ISBN 8–29191–708–6
- [Dör87] DÖRNER, D.: *Problemlösen als Informationsverarbeitung*. 3. Aufl. Stuttgart, Berlin, Köln, Mainz: Kohlhammer, 1987. ISBN 3–17009–711–3
- [DPAM00] DEB, K.; PRATAP, A.; AGARWAL, S.; MEYARIVAN, T.: *A Fast and Elitist Multi-Objective Genetic Algorithm: NSGA-II*. Tech. Report, Kanpur Genetic Algorithms Laboratory (KanGAL), Report Number 2000001, 2000
- [DPAM02] DEB, K.; PRATAP, A.; AGARWAL, S.; MEYARIVAN, T.: *A Fast and Elitist Multiobjective Genetic Algorithm: NSGA-II*. In: *IEEE Transactions on Evolutionary Computation* 6 (2002), Nr. 2, S. 182–197. DOI 10.1109/4235.996017
- [DW16] DHONDT, G.; WITTIG, K.: *CalculiX: A Free Software Three-Dimensional Structural Finite Element Program*. <http://www.calculix.de/>, Abruf: 07.07.2016
- [DZ13] DUDDECK, F.; ZIMMER, H.: *Modular Car Body Design and Optimization by an Implicit Parameterization Technique via SFE CONCEPT*. In: *Proceedings of the FISITA 2012 World Automotive Congress*. 195. Berlin, Heidelberg: Springer-Verlag, 2013, S. 413–424. ISBN 978–3–642–33834–2. DOI 10.1007/978-3-642-33835-9_39

- [Dzi15] DZIUBACZYK, B.: *Evaluieren von Methoden und Strategien zum verteilten Rechnen von rechenintensiven Problemen in der Produktentwicklung und prototypische Umsetzung*. Masterarbeit. Otto-von-Guericke-Universität Magdeburg, Magdeburg, 2015
- [EBS15] ERNST, H.; BENEKEN, G. H.; SCHMIDT, J.: *Grundkurs Informatik: Grundlagen und Konzepte für die erfolgreiche IT-Praxis - eine umfassende, praxisorientierte Einführung*. 5., vollst. überarb. Aufl. Wiesbaden: Springer Vieweg, 2015. ISBN 978-3-65801-627-2
- [ECSL14] EGAN, P. F.; CAGAN, J.; SCHUNN, C.; LEDUC, P. R.: *Cognitive-Based Search Strategies for Complex Bio-Nanotechnology Design Derived Through Symbiotic Human and Agent-Based Approaches*. In: *ASME 2014 International Design Engineering Technical Conferences and Computers and Information in Engineering Conference*. Buffalo, NY, 17.–20.08.2014. New York: American Society of Mechanical Engineers, ISBN 978-0-7918-4640-7. DOI 10.1115/DETC2014-34714
- [ECSL15] EGAN, P.; CAGAN, J.; SCHUNN, C.; LEDUC, P.: *Synergistic human-agent methods for deriving effective search strategies: the case of nanoscale design*. In: *Research in Engineering Design* 26 (2015), Nr. 2, S. 145–169. ISSN 0934-9839. DOI 10.1007/s00163-015-0190-3
- [EGH⁺15] ELLSON, J.; GANSNER, E.; HU, Y.; BILGIN, A.; PERRY, D.: *Graphviz - Graph Visualization Software*. <http://www.graphviz.org/>, Abruf: 04.06.2015
- [Ehr95] EHRENSPIEL, K.: *Integrierte Produktentwicklung: Methoden für Prozeßorganisation, Produkterstellung und Konstruktion*. München, Wien: Hanser, 1995. ISBN 3-446-15706-9
- [Ehr05] EHRGOTT, M.: *Multicriteria Optimization*. 2nd ed. Berlin, New York: Springer, 2005. ISBN 3-540-21398-8
- [EKLM14] EHRENSPIEL, K.; KIEWERT, A.; LINDEMANN, U.; MÖRTL, M.: *Kostengünstig Entwickeln und Konstruieren: Kostenmanagement bei der integrierten Produktentwicklung*. 7. Aufl. Berlin, Heidelberg: Springer-Verlag, 2014. ISBN 978-3-64241-958-4
- [EKS94] ESCHENAUER, H. A.; KOBELEV, V. V.; SCHUMACHER, A.: *Bubble method for topology and shape optimization of structures*. In: *Structural Optimization* 8 (1994), Nr. 1, S. 42–51. ISSN 0934-4373. DOI 10.1007/BF01742933
- [Esc85] ESCHENAUER, H.: *Rechnerische und experimentelle Untersuchungen zur Strukturoptimierung von Bauweisen: Ein Forschungsvorhaben der Deutschen Forschungsgemeinschaft (DFG)*. Universität-Gesamthochschule Siegen. 1985
- [FCK10] FU, K.; CAGAN, J.; KOTOVSKY, K.: *Design Team Convergence: The Influence of Example Solution Quality*. In: *Journal of Mechanical Design* 132 (2010), Nr. 11, S. 111005. DOI 10.1115/1.4002202
- [Fen14] FENDER, J. H. W.: *Solution Spaces for Vehicle Crash Design*. Dissertation. Technische Universität München, München, 2014

- [FF93] FONSECA, C. M.; FLEMING, P. J.: *Genetic Algorithms for Multiobjective Optimization: Formulation, Discussion and Generalization*. In: FORREST, S. (Hrsg.): *Proceedings of the Fifth International Conference on Genetic Algorithms*. Urbana-Champaign, IL, 17.–21.07.1993. San Mateo, Calif.: Morgan Kaufmann Publishers, S. 416–423. ISBN 1–55860–299–2
- [FF98] FONSECA, C. M.; FLEMING, P. J.: *Multiobjective Optimization and Multiple Constraint Handling with Evolutionary Algorithms. I. A Unified Formulation*. In: *IEEE Transactions on Systems, Man, and Cybernetics - Part A: Systems and Humans* 28 (1998), Nr. 1, S. 26–37. DOI 10.1109/3468.650319
- [FFBH12] FIRL, M.; FISCHER, M.; BLETZINGER, K.-U.; HARZHEIM, L.: *Innovative Formoptimierung in der Fahrzeugentwicklung*. In: *ATZ - Automobiltechnische Zeitschrift* 114 (2012), Nr. 3, S. 224–229. ISSN 0001–2785. DOI 10.1365/s35148-012-0292-4
- [FG13] FELDHUSEN, J.; GROTE, K.-H. (Hrsg.): *Pahl/Beitz Konstruktionslehre: Methoden und Anwendung erfolgreicher Produktentwicklung*. 8., vollst. überarb. Aufl. Berlin, Heidelberg: Springer-Verlag, 2013. ISBN 978–3–642–29568–3
- [FGLSV13] FERREIRO, A. M.; GARCÍA, J. A.; LÓPEZ-SALAS, J. G.; VÁZQUEZ, C.: *An efficient implementation of parallel simulated annealing algorithm in GPUs*. In: *Journal of Global Optimization* 57 (2013), Nr. 3, S. 863–890. DOI 10.1007/s10898-012-9979-z
- [FH11] FISCHER, P.; HOFER, P.: *Lexikon der Informatik*. 15., überarb. Aufl. Berlin, Heidelberg: Springer-Verlag, 2011. ISBN 978–3–64215–125–5
- [FK09] FORRESTER, A. I.; KEANE, A. J.: *Recent advances in surrogate-based optimization*. In: *Progress in Aerospace Sciences* 45 (2009), Nr. 1–3, S. 50–79. ISSN 0376–0421. DOI 10.1016/j.paerosci.2008.11.001
- [FKBV12] FRITZSCHE, M.; KITTEL, K.; BLANKENBURG, A.; VAJNA, S.: *Multidisciplinary design optimisation of a recurve bow based on applications of the autogenetic design theory and distributed computing*. In: *Enterprise Information Systems* 6 (2012), Nr. 3, S. 329–343. ISSN 1751–7575. DOI 10.1080/17517575.2011.650216
- [FKST98] FELDMANN, A.; KAO, M.-Y.; SGALL, J.; TENG, S.-H.: *Optimal On-Line Scheduling of Parallel Jobs with Dependencies*. In: *Journal of Combinatorial Optimization* 1 (1998), Nr. 4, S. 393–411. DOI 10.1023/A:1009794729459
- [Fle64] FLETCHER, R.: *Function minimization by conjugate gradients*. In: *The Computer Journal* 7 (1964), Nr. 2, S. 149–154. ISSN 0010–4620. DOI 10.1093/comjnl/7.2.149
- [Fle70] FLETCHER, R.: *A New Approach to Variable Metric Algorithms*. In: *The Computer Journal* 13 (1970), Nr. 3, S. 317–322. ISSN 0010–4620. DOI 10.1093/comjnl/13.3.317
- [Fon95] FONSECA, C. M.: *Multiobjective Genetic Algorithms with Application to Control Engineering Problems*. Dissertation. University of Sheffield, Sheffield, UK, 1995

- [FOW66] FOGEL, L. J.; OWENS, A. J.; WALSH, M. J.: *Artificial Intelligence through Simulated Evolution*. New York: John Wiley & Sons, 1966
- [FP63] FLETCHER, R.; POWELL, M. J. D.: *A Rapidly Convergent Descent Method for Minimization*. In: *The Computer Journal* 6 (1963), Nr. 2, S. 163–168. ISSN 0010–4620. DOI 10.1093/comjnl/6.2.163
- [Fre09] FREE SOFTWARE FOUNDATION, INC.: *GNUbatch*. <https://www.gnu.org/software/gnubatch/>, Abruf: 21.12.2015
- [Fün06] FÜNDELING, C.-U.: *Ressourcenbeschränkte Projektplanung bei vorgegebenen Arbeitsvolumina*. Dissertation. Universität Karlsruhe, Karlsruhe, 2006
- [Gab16] GABLER (Hrsg.): *Gabler Wirtschaftslexikon, Stichwort: vollständige Enumeration*. <http://wirtschaftslexikon.gabler.de/Archiv/122402/vollstaendige-enumeration-v5.html>, Abruf: 14.05.2016
- [Gel14] GELDERMANN, J.: *Multikriterielle Optimierung*. <http://www.enzyklopaedie-der-wirtschaftsinformatik.de/lexikon/technologien-methoden/Operations-Research/Mathematische-Optimierung/Multikriterielle-Optimierung>, Abruf: 24.08.2016
- [Gen15] GENTZSCH, W.: *Smart Manufacturing: CAE as a Service, in the Cloud*. In: *10th European LS-DYNA Conference*. Würzburg, 15.–17.06.2015
- [GH02] GROENWOLD, A. A.; HINDLEY, M. P.: *Competing parallel algorithms in structural optimization*. In: *Structural and Multidisciplinary Optimization* 24 (2002), Nr. 5, S. 343–350. ISSN 1615–147X. DOI 10.1007/s00158-002-0246-7
- [GKK04] GERDES, I.; KLAWONN, F.; KRUSE, R.: *Evolutionäre Algorithmen: Genetische Algorithmen - Strategien und Optimierungsverfahren - Beispielanwendungen*. Wiesbaden: Friedr. Vieweg & Sohn Verlag/GWV Fachverlage GmbH, 2004. ISBN 3–52805–570–7
- [Gol70] GOLDFARB, D.: *A Family of Variable-Metric Methods Derived by Variational Means*. In: *Mathematics of Computation* 24 (1970), Nr. 109, S. 23. DOI 10.1090/S0025-5718-1970-0258249-6
- [Gol89] GOLDBERG, D. E.: *Genetic Algorithms in Search, Optimization, and Machine Learning*. Reading, MA: Addison-Wesley Publishing Company, Inc., 1989. ISBN 0–201–15767–5
- [Gor14] GORDON, R.: *Oddjob*. <http://rgordon.co.uk/oddjob/index.html>, Abruf: 21.12.2015
- [GP07] GUTIN, G.; PUNNEN, A. P.: *Combinatorial Optimization*. Bd. 12: *The Traveling Salesman Problem and Its Variations*. Boston, MA: Springer US, 2007. ISBN 978–0–387–44459–8
- [GR77] GABRIELE, G. A.; RAGSDALL, K. M.: *The Generalized Reduced Gradient Method: A Reliable Tool for Optimal Design*. In: *Journal of Engineering for Industry* 99 (1977), Nr. 2, S. 394. DOI 10.1115/1.3439249
- [Gri13] GRID ENGINE: *Open Grid Scheduler*. <http://gridscheduler.sourceforge.net/index.html>, Abruf: 21.12.2015

- [Gro14] GROSSKLAUS, R. H. G.: *Von der Produktidee zum Markterfolg: Innovationen planen, einführen und erfolgreich managen*. 2. Auflage. Wiesbaden: Springer Gabler, 2014. ISBN 978-3-8349-4593-8
- [GSSR15] GLYMOUR, C.; SCHEINES, R.; SPIRITES, P.; RAMSEY, J.: *The TETRAD Project: Causal Models and Statistical Data*. <http://www.phil.cmu.edu/projects/tetrad/>, Abruf: 04.06.2015
- [GU14] GARDNER, A.; UNGER, J.: *Depends: Workflow Management Software for Visual Effects Production*. In: *Proceedings of the Fourth Symposium on Digital Production*. Vancouver, BC, Kanada, 09.08.2014. New York: ACM, S. 19–24. ISBN 978-1-45033-044-2
- [Har14] HARZHEIM, L.: *Strukturoptimierung: Grundlagen und Anwendungen*. 2., überarb. und erw. Aufl. Haan-Gruiten: Europa-Lehrmittel, 2014. ISBN 978-3-80855-659-7
- [Häu11] HÄUSLER, O.: *Business-Impact-Management von Informationstechnologie im Unternehmen: Geschäftsprozessorientierte Planung, Steuerung und Kontrolle der IT*. Dissertation. Justus-Liebig-Universität Gießen, Gießen, 2011
- [HC15] HEINE, A.; CPU 24/7: *Cloud-Technologie: Leistung auf Abruf*. In: *PLM IT REPORT 5* (2015), S. 26–285. ISSN 0930-7117
- [HCB05] HUGHES, T.; COTTRELL, J. A.; BAZILEVS, Y.: *Isogeometric analysis: CAD, finite elements, NURBS, exact geometry and mesh refinement*. In: *Computer Methods in Applied Mechanics and Engineering* 194 (2005), Nr. 39-41, S. 4135–4195. ISSN 0045-7825. DOI 10.1016/j.cma.2004.10.008
- [HDF12] HWANG, K.; DONGARRA, J. J.; FOX, G. C.: *Distributed and Cloud Computing: From Parallel Processing to the Internet of Things*. Amsterdam, London: Elsevier/Morgan Kaufmann, 2012. ISBN 978-0-12385-880-1
- [HG05] HARZHEIM, L.; GRAF, G.: *A review of optimization of cast parts using topology optimization*. In: *Structural and Multidisciplinary Optimization* 30 (2005), Nr. 6, S. 491–497. ISSN 1615-147X. DOI 10.1007/s00158-005-0553-x
- [HG06] HARZHEIM, L.; GRAF, G.: *A review of optimization of cast parts using topology optimization*. In: *Structural and Multidisciplinary Optimization* 31 (2006), Nr. 5, S. 388–399. ISSN 1615-147X. DOI 10.1007/s00158-005-0554-9
- [HK04] HENZE, S.; KÖHNE, K.: *Condor, Condor-G, Classad*. In: TRÖGER, P.; WAGNER, S. (Hrsg.): *Grid computing*. Technische Berichte des Hasso-Plattner-Instituts für Softwaresystemtechnik an der Universität Potsdam, Nr. 3. Potsdam: Universitätsverlag, 2004, ISBN 3-937786-28-7
- [HL92] HAJELA, P.; LIN, C.-Y.: *Genetic Search Strategies in Multicriterion Optimal Design*. In: *Structural Optimization* 4 (1992), Nr. 2, S. 99–107. ISSN 0934-4373. DOI 10.1007/BF01759923
- [HLF13] HOFMANN, J.; LIMMER, S.; FEY, D.: *Performance investigations of genetic algorithms on graphics cards*. In: *Swarm and Evolutionary Computation* 12 (2013), S. 33–47. ISSN 2210-6502. DOI 10.1016/j.swevo.2013.04.003

- [HNG94] HORN, J.; NAFPLIOTIS, N.; GOLDBERG, D. E.: *A Niched Pareto Genetic Algorithm for Multiobjective Optimization*. In: *First IEEE Conference on Evolutionary Computation. IEEE World Congress on Computational Intelligence*. Orlando, FL, USA, 27.–29.06.1994, IEEE, S. 82–87. ISBN 0–7803–1899–4. DOI 10.1109/ICEC.1994.350037
- [HOG95] HANSEN, N.; OSTERMEIER, A.; GAWELCZYK, A.: *On the Adaptation of Arbitrary Normal Mutation Distributions in Evolution Strategies: The Generating Set Adaptation*. In: ESHELMAN, L. J. (Hrsg.): *Proceedings of the Sixth International Conference on Genetic Algorithms*. Pittsburgh, PA, 15.–19.07.1995. San Francisco: Kaufman, ISBN 1–558–60370–0
- [Hol75] HOLLAND, J. H.: *Adaptation in Natural and Artificial Systems: An Introductory Analysis with Applications to Biology, Control, and Artificial Intelligence*. The University of Michigan Press, 1975
- [HTK13] HOLMBERG, E.; TORSTENFELT, B.; KLARBRING, A.: *Stress constrained topology optimization*. In: *Structural and Multidisciplinary Optimization* 48 (2013), Nr. 1, S. 33–47. ISSN 1615–147X. DOI 10.1007/s00158-012-0880-7
- [Hu61] HU, T. C.: *Parallel Sequencing and Assembly Line Problems*. In: *Operations Research* 9 (1961), Nr. 6, S. 841–848. ISSN 0030–364X. DOI 10.1287/opre.9.6.841
- [Ind15] INDEPENDIT INTEGRATIVE TECHNOLOGIES GMBH: *schedulix*. <http://www.schedulix.org/de>, Abruf: 21.12.2015
- [Ins12] INSPIRED INTELLIGENCE INITIATIVE: *Inspyred: Bio-inspired Algorithms in Python*. <http://pythonhosted.org/inspyred/>, Abruf: 18.04.2016
- [JA93] JAIN, P.; AGOGINO, A. M.: *Global Optimization Using the Multistart Method*. In: *Journal of Mechanical Design* 115 (1993), Nr. 4, S. 770–775
- [JC04] JORDAN, A.; CLEMENT, S.: *Handbuch für das Konstruktionssystem NOA*. Lehrstuhl für Maschinenbauinformatik, Otto-von-Guericke-Universität Magdeburg, 2004
- [JGG⁺11] JIA, X.; GU, X.; GRAVES, Y. J.; FOLKERTS, M.; JIANG, S. B.: *GPU-based fast Monte Carlo simulation for radiotherapy dose calculation*. In: *Physics in medicine and biology* 56 (2011), Nr. 22, S. 7017–7031. ISSN 1361–6560. DOI 10.1088/0031-9155/56/22/002
- [JSP15] JSPLUMB PTY LTD.: *JSPlumb Toolkit: Build Connectivity Fast*. <https://jsplumbtoolkit.com/>, Abruf: 04.06.2015
- [Kah03] KAHN, C.: *Supercomputer "Big Mac": Der dickste Apfel aller Zeiten*. <http://www.spiegel.de/netzwelt/tech/supercomputer-big-mac-der-dickste-apfel-aller-zeiten-a-274441.html>, Abruf: 14.01.2016
- [Kan08] KANPUR GENETIC ALGORITHMS LABORATORY: *Software Developed at KanGAL*. <http://www.iitk.ac.in/kangal/codes.shtml>, Abruf: 18.04.2016

- [KBB⁺15] KRUSE, R.; BORGELT, C.; BRAUNE, C.; KLAWONN, F.; MOEWES, C.; STEINBRECHER, M.: *Computational Intelligence: Eine methodische Einführung in Künstliche Neuronale Netze, Evolutionäre Algorithmen, Fuzzy-Systeme und Bayes-Netze*. 2., überarb. und erw. Aufl. Wiesbaden: Springer Vieweg, 2015 (Lehrbuch). ISBN 978-3-658-10903-5
- [KC00] KNOWLES, J. D.; CORNE, D. W.: *Approximating the Nondominated Front Using the Pareto Archived Evolution Strategy*. In: *Evolutionary Computation* 8 (2000), Nr. 2, S. 149–172. ISSN 1063-6560. DOI 10.1162/106365600568167
- [KE95] KENNEDY, J.; EBERHART, R.: *Particle Swarm Optimization*. In: *International Conference on Neural Networks (ICNN'95)*. Perth, Australien, 27.11.–01.12.1995, S. 1942–1948. ISBN 0-7803-2768-3. DOI 10.1109/ICNN.1995.488968
- [Kep94] KEPHART, J. O.: *A Biologically Inspired Immune System for Computers*. In: BROOKS, R. A.; MAES, P. (Hrsg.): *Artificial life IV*. Cambridge, MA: MIT Press, S. 130–139. ISBN 978-0-58503-839-1
- [KFV11] KITTEL, K.; FRITZSCHE, M.; VAJNA, S.: *The autogenetic design theory and its practical application*. In: *International Journal of Design Engineering* 4 (2011), Nr. 2, S. 91–113. DOI 10.1504/IJDE.2011.045130
- [KG06] KINZLER, S.; GRABE, J.: *Wirtschaftliche Optimierung rückverankerter Spundwandkonstruktionen*. In: GRABE, J. (Hrsg.): *Optimierung in der Geotechnik*. Hamburg, 10.10.2006, Technische Universität Hamburg-Harburg, Institut für Geotechnik und Baubetrieb, ISBN 3-93631-013-0
- [KGV83] KIRKPATRICK, S.; GELATT, C. D.; VECCHI, M. P.: *Optimization by Simulated Annealing*. In: *Science* 220 (1983), Nr. 4598, S. 671–680. DOI 10.1126/science.220.4598.671
- [KHL⁺11] KUSTER, J.; HUBER, E.; LIPPMANN, R.; SCHMID, A.; SCHNEIDER, E.; WITSCHI, U.; WÜST, R.: *Handbuch Projektmanagement*. Berlin, Heidelberg: Springer-Verlag, 2011. ISBN 978-3-642-21242-0
- [KHVZ11] KITTEL, K.; HEHENBERGER, P.; VAJNA, S.; ZEMAN, K.: *Product Model of the Autogenetic Design Theory*. In: CULLEY, S. J.; HICKS, B. J.; MCALOONE, T. C.; HOWARD, T. J.; LINDEMANN, U. (Hrsg.): *Proceedings of the 18th International Conference on Engineering Design (ICED 11)*. Kopenhagen, Dänemark, 15.–18.08.2011, ISBN 978-1-904670-24-7
- [Kim01] KIM, H. M.: *Target Cascading in Optimal System Design*. Dissertation. University of Michigan, Ann Arbor, MI, 2001
- [Kin11] KINZLER, S.: *Zur Parameteridentifikation, Entwurfs- und Strukturoptimierung in der Geotechnik mittels numerischer Verfahren*. Dissertation. Technische Universität Hamburg-Harburg, Hamburg, 2011
- [KMKM09] KUREICHIK, V. M.; MALIOUKOV, S. P.; KUREICHIK, V. V.; MALIOUKOV, A. S.: *Studies in Computational Intelligence*. Bd. 212: *Genetic Algorithms for Applied CAD Problems*. Berlin, Heidelberg: Springer-Verlag, 2009. ISBN 978-3-54085-280-3

- [KMPJ03] KIM, H. M.; MICHELENA, N. F.; PAPALAMBROS, P. Y.; JIANG, T.: *Target Cascading in Optimal System Design*. In: *Journal of Mechanical Design* 125 (2003), Nr. 3, S. 474. DOI 10.1115/1.1582501
- [Knü11] KNÜPPEL, S.: *DAG program*. <http://epi.dife.de/dag/>, Abruf: 04.06.2015
- [Kri51] KRIGE, D. G.: *A Statistical Approach to Some Basic Mine Valuation Problems on the Witwatersrand*. Masterarbeit. University of the Witwatersrand, Johannesburg, 1951
- [KTAG10] KAMPOLIS, I. C.; TROMPOUKIS, X. S.; ASOUTI, V. G.; GIANNAKOGLU, K. C.: *CFD-based analysis and two-level aerodynamic optimization on graphics processing units*. In: *Computer Methods in Applied Mechanics and Engineering* 199 (2010), Nr. 9-12, S. 712–722. ISSN 0045–7825. DOI 10.1016/j.cma.2009.11.001
- [Kun05] KUNKLE, D.: *A Summary and Comparison of MOEA Algorithms*. Northeastern University, Boston, MA, 2005
- [Kun15] KUNZMANN, N.: *Nullege: FlowchartCustomNode*. <http://nullege.com/codes/show/src%40p%40y%40pyqtgraph-0.9.8%40examples%40FlowchartCustomNode.py/9/pyqtgraph.flowchart.Flowchart/python>, Abruf: 04.06.2015
- [KV16] KUMAR, N.; VIDYARTHI, D. P.: *A novel hybrid PSO–GA meta-heuristic for scheduling of DAG with communication on multiprocessor systems*. In: *Engineering with Computers* 32 (2016), Nr. 1, S. 35–47. ISSN 0177–0667. DOI 10.1007/s00366-015-0396-z
- [KWW15] KÜSTNER, C.; WACHSMUTH, P.; WARTZACK, S.: *Datenakquisition und Analyse im Assistenzsystem zur lärmreduzierten Auslegung rotierender Maschinen*. In: BINZ, H.; BERTSCHE, B.; BAUER, W.; ROTH, D. (Hrsg.): *Beiträge zum Stuttgarter Symposium für Produktentwicklung (SSP2015)*. 19.06.2015. Stuttgart: Fraunhofer IAO
- [KY16] KANG, P.; YOUN, S.-K.: *Isogeometric shape optimization of trimmed shell structures*. In: *Structural and Multidisciplinary Optimization* 53 (2016), Nr. 4, S. 825–845. ISSN 1615–147X. DOI 10.1007/s00158-015-1361-6
- [LB09] LEHMANN, S.; BUXMANN, P.: *Preisstrategien von Softwareanbietern*. In: *Wirtschaftsinformatik* 51 (2009), Nr. 6, S. 519–529. DOI 10.1007/s11576-009-0197-3
- [LCF15] LI, S.-S.; CHEN, R.-X.; FENG, Q.: *Scheduling two job families on a single machine with two competitive agents*. In: *Journal of Combinatorial Optimization* (2015). DOI 10.1007/s10878-015-9902-x
- [Lee05] LEE, S. H.: *A CAD–CAE integration approach using feature-based multi-resolution and multi-abstraction modelling techniques*. In: *Computer-Aided Design* 37 (2005), Nr. 9, S. 941–955. ISSN 0010–4485. DOI 10.1016/j.cad.2004.09.021
- [Lee13] LEE, R. Y.: *Software Engineering: A Hands-On Approach*. Atlantis Press, 2013. ISBN 978–9–46239–005–8

- [Lee14] LEE, A.: *pyDOE: The experimental design package for python*. <https://pythonhosted.org/pyDOE/>, Abruf: 17.12.2015
- [Lim10] LIMMER, S.: *Condor*. In: FEY, D. (Hrsg.): *Grid-Computing*. Berlin, Heidelberg: Springer-Verlag, 2010, S. 311–321. ISBN 978–3–54079–746–3
- [Lin14] LINKÖPING UNIVERSITY: *Depends Workflow Management*. <http://www.dependsworkflow.net/index.html>, Abruf: 04.06.2015
- [LM12] LAMBE, A. B.; MARTINS, JOAQUIM R. R. A.: *Extensions to the design structure matrix for the description of multidisciplinary design, analysis, and optimization processes*. In: *Structural and Multidisciplinary Optimization* 46 (2012), Nr. 2, S. 273–284. ISSN 1615–147X. DOI 10.1007/s00158-012-0763-y
- [LRL03] LIN, C.; ROERMUND, ARTHUR H. M. VAN; LEENAERTS, DOMINE M. W.: *Kluwer international series in engineering and computer science*. Bd. 751. *Analog Circuits and Signal Processing: Mixed-Signal Layout Generation Concepts*. New York, Boston, Dordrecht, London, Moscow: Kluwer Academic Publishers, 2003. ISBN 1–40207–598–7
- [LSSY10] LEYVAND, Y.; SHABTAY, D.; STEINER, G.; YEDIDSON, L.: *Just-in-time scheduling with controllable processing times on parallel machines*. In: *Journal of Combinatorial Optimization* 19 (2010), Nr. 3, S. 347–368. DOI 10.1007/s10878-009-9270-5
- [MA04] MARLER, R. T.; ARORA, J. S.: *Survey of multi-objective optimization methods for engineering*. In: *Structural and Multidisciplinary Optimization* 26 (2004), Nr. 6, S. 369–395. ISSN 1615–147X. DOI 10.1007/s00158-003-0368-6
- [Ma07] MA, L.; DESIGN COUNCIL (Hrsg.): *A study of the design process: Eleven lessons: managing design in eleven global brands*. <http://www.designcouncil.org.uk/resources/report/11-lessons-managing-design-global-brands>, Abruf: 14.10.2016
- [MAAE+95] MOLINA, A.; AL-ASHAAB, A.; ELLIS, T.; YOUNG, R.; BELL, R.: *A Review of Computer-Aided Simultaneous Engineering Systems*. In: *Research in Engineering Design* 7 (1995), Nr. 1, 38–63. ISSN 0934–9839. DOI 10.1007/BF01681911
- [Maj14] MAJIĆ, N.: *Entwicklung einer FEM-basierten Methode zur fertigungsorientierten Sickenmustergestaltung für biegebeanspruchte Tragstrukturen*. Dissertation. Karlsruher Institut für Technologie, Karlsruhe, 2014
- [MC16] MÜLLER, A.; CPU 24/7: *Simulieren ohne Grenzen*. In: *PLM IT REPORT* 1 (2016), S. 18–21. ISSN 0930–7117
- [MCK15a] MCCOMB, C.; CAGAN, J.; KOTOVSKY, K.: *Heterogeneous Simulated Annealing Teams: An Optimizing Search Algorithm Inspired by Engineering Design Teams*. In: WEBER, C.; HUSUNG, S.; CASCINI, G.; CANTAMESSA, M.; MARJANOVIĆ, D. (Hrsg.): *Proceedings of the 20th International Conference on Engineering Design (ICED 15), Vol. 6: Design Methods and Tools - Part 2*. Mailand, Italien, 27.–30.07.2015, S. 225–234. ISBN 978–1–90467–069–8

- [MCK15b] McCOMB, C.; CAGAN, J.; KOTOVSKY, K.: *Rolling with the punches: An examination of team performance in a design task subject to drastic changes*. In: *Design Studies* 36 (2015), S. 99–121. ISSN 0142694X. DOI 10.1016/j.destud.2014.10.001
- [Men32] MENGER, K.: *Das Botenproblem*. In: *Ergebnisse eines mathematischen Kolloquiums* 2 (1932), S. 11–12
- [MGP60] MILLER, G. A.; GALANTER, E.; PRIBRAM, K. H.: *Plans and the Structure of Behavior*. New York: Holt, 1960. ISBN 0-03010-075-5
- [MGP91] MILLER, G. A.; GALANTER, E.; PRIBRAM, K. H.: *Strategien des Handelns: Pläne und Strukturen des Verhaltens*. 2. Aufl. Stuttgart: Klett-Cotta, 1991. ISBN 3-60895-102-4
- [Mic16] MICROSOFT: *Ruhezustände für Wake-On-LAN*. <https://technet.microsoft.com/de-de/library/bb693821.aspx>, Abruf: 09.02.2016
- [ML13] MARTINS, JOAQUIM R. R. A.; LAMBE, A. B.: *Multidisciplinary Design Optimization: A Survey of Architectures*. In: *AIAA Journal* 51 (2013), Nr. 9, S. 2049–2075. ISSN 0001-1452. DOI 10.2514/1.J051895
- [MLV⁺09] MEYER, O.; LAUER, I.; VOLZ, K.; SUHRER, A.; BAIER, H.: *Parametrischer Geometrieprozess und Funktionsauslegung in frühen Entwicklungsphasen*. In: STARK, R.; RÖTH, T. (Hrsg.): *Karosseriekongress Systemintegration in der Karosserietechnik*. Baden-Baden, 17.–18.03.2009. Düsseldorf: VDI-Wissensforum, ISBN 978-3-981-26244-5
- [MMAC09] MYERS, R. H.; MONTGOMERY, D. C.; ANDERSON-COOK, C. M.: *Response Surface Methodology: Process and Product Optimization Using Designed Experiments*. 3. Aufl. Hoboken, NJ: John Wiley & Sons, 2009. ISBN 978-0-470-17446-3
- [Mor78] MORÉ, J. J.: *The Levenberg-Marquardt Algorithm: Implementation and Theory*. In: WATSON, G. A. (Hrsg.): *Numerical Analysis*. Berlin, Heidelberg: Springer-Verlag, 1978, S. 105–116. ISBN 978-3-540-08538-6. DOI 10.1007/BFb0067700
- [Mos10] MOSTAGHIM, S.: *Parallel Multi-objective Optimization Using Self-organized Heterogeneous Resources*. In: KACPRZYK, J.; VEGA, F. F.; CANTÚ-PAZ, E. (Hrsg.): *Parallel and Distributed Computational Intelligence*. 269. Berlin, Heidelberg: Springer-Verlag, 2010, S. 165–179. ISBN 978-3-642-10674-3. DOI 10.1007/978-3-642-10675-0_8
- [MPR15] MÜNZL, G.; PAULY, M.; RETI, M.: *Cloud Computing als neue Herausforderung für Management und IT*. Berlin, Heidelberg: Springer-Verlag, 2015. ISBN 978-3-662-45831-0
- [MRR⁺53] METROPOLIS, N.; ROSENBLUTH, A. W.; ROSENBLUTH, M. N.; TELLER, A. H.; TELLER, E.: *Equation of State Calculations by Fast Computing Machines*. In: *The Journal of Chemical Physics* 21 (1953), Nr. 6, S. 1087. DOI 10.1063/1.1699114

- [MS11] MISHRA, V.; SURESH, K.: *GPU-Friendly Preconditioners for Efficient 3-D Finite Element Analysis of Thin Structures*. In: *ASME 2011 International Design Engineering Technical Conferences and Computers and Information in Engineering Conference*. Washington, DC, 28.–31.08.2011. New York: American Society of Mechanical Engineers, S. 339–346. ISBN 978–0–791–85479–2. DOI 10.1115/DETC2011-47330
- [MSV93] MÜHLENBEIN, H.; SCHLIERKAMP-VOOSEN, D.: *Predictive Models for the Breeder Genetic Algorithm: I. Continuous Parameter Optimization*. In: *Evolutionary Computation* 1 (1993), Nr. 1, S. 25–49. ISSN 1063–6560. DOI 10.1162/evco.1993.1.1.25
- [MSV95] MÜHLENBEIN, H.; SCHLIERKAMP-VOOSEN, D.: *Analysis of Selection, Mutation and Recombination in Genetic Algorithms*. In: BANZHAF, W.; EECKMAN, F. H. (Hrsg.): *Evolution and Biocomputation*. Berlin, New York: Springer, 1995, ISBN 3–540–59046–3
- [Müh91] MÜHLENBEIN, H.: *Evolution in Time and Space - The Parallel Genetic Algorithm*. In: RAWLINS, G. J. E. (Hrsg.): *Foundations of Genetic Algorithms*. San Mateo, CA: Morgan Kaufmann, 1991, S. 316–337. ISBN 978–1–558–60170–3
- [Müh92] MÜHLENBEIN, H.: *Parallel Genetic Algorithm in Combinatorial Optimization*. In: BALCI, O.; SHARDA, R.; ZENIOS, S. A. (Hrsg.): *Computer Science and Operations Research*. Oxford, New York: Pergamon Press, 1992, S. 441–456. ISBN 0–08–040806–0
- [Müh94] MÜHLENBEIN, H.: *The Breeder Genetic Algorithm-A Provable Optimal Search Algorithm and its Application*. In: *IEE Colloquium on Applications of Genetic Algorithms*. London: Institution of Electrical Engineers, S. 5/1–5/3
- [Nas51] NASH, J.: *Non-Cooperative Games*. In: *The Annals of Mathematics* 54 (1951), Nr. 2, S. 286–295. ISSN 0003–486X. DOI 10.2307/1969529
- [Net14] NETWORKX DEVELOPER TEAM: *NetworkX: High-productivity software for complex networks*. <https://networkx.github.io/index.html>, Abruf: 04.06.2015
- [NL05] NITSOPOULOS, I.; LAUBER, B.: *Multidisziplinäre Optimierung mit OPTIMUS unter Berücksichtigung von Crash, NVH und Lebensdauer*. In: *4. Deutsches LS-DYNA Forum*. Bamberg, 20.–21.10.2005
- [NM65] NELDER, J. A.; MEAD, R.: *A Simplex Method for Function Minimization*. In: *The Computer Journal* 7 (1965), Nr. 4, S. 308–313. ISSN 0010–4620. DOI 10.1093/comjnl/7.4.308
- [NW89] NEVINS, J. L.; WHITNEY, D. E.: *Concurrent Design of Products and Processes: A Strategy For the Next Generation in Manufacturing*. New York: McGraw-Hill, 1989. ISBN 978–0–07046–341–7

- [OGH94] OSTERMEIER, A.; GAWELCZYK, A.; HANSEN, N.: *Step-Size Adaptation Based on Non-Local Use of Selection Information*. In: GOOS, G.; HARTMANIS, J.; LEEUWEN, J.; DAVIDOR, Y.; SCHWEFEL, H.-P.; MÄNNER, R. (Hrsg.): *Parallel Problem Solving from Nature PPSN III*. Berlin, Heidelberg: Springer, 1994, S. 189–198. ISBN 978-3-540-58484-1. DOI 10.1007/3-540-58484-6_263
- [PB46] PLACKETT, R. L.; BURMAN, J. P.: *The Design of Optimum Multifactorial Experiments*. In: *Biometrika* 33 (1946), Nr. 4, S. 305–325. ISSN 0006-3444. DOI 10.2307/2332195
- [PDT⁺96] PAWAR, K. S.; DRIVA, H.; THOBEN, K.-D.; OHELMANN, R.; WEBER, F.: *Concurrent Engineering: From Concept to Implementation*. In: CHOWDIAH, M. P.; GARGESA, G.; ARUN KUMAR, V. (Hrsg.): *Agile Manufacturing*. Bangalore, Indien, 22.–24.02.1996. New Delhi, New York: Tata-McGraw-Hill Pub. Co., ISBN 978-0-07463-052-5
- [Ped00] PEDERSEN, P.: *On optimal shapes in materials and structures*. In: *Structural and Multidisciplinary Optimization* 19 (2000), Nr. 3, S. 169–182. ISSN 1615-147X. DOI 10.1007/s001580050100
- [Pfl98] PFLEEGER, S. L.: *Software Engineering: Theory and Practice*. Upper Saddle River, NJ: Prentice Hall, 1998. ISBN 0-13-624842-X
- [Poh00] POHLHEIM, H.: *Evolutionäre Algorithmen: Verfahren, Operatoren und Hinweise für die Praxis*. Berlin, Heidelberg: Springer-Verlag, 2000. ISBN 978-3-642-63052-1
- [Pow64] POWELL, M. J. D.: *An efficient method for finding the minimum of a function of several variables without calculating derivatives*. In: *The Computer Journal* 7 (1964), Nr. 2, S. 155–162. ISSN 0010-4620. DOI 10.1093/comjnl/7.2.155
- [Pow87] POWELL, M. J. D.: *Radial Basis Functions for Multivariable Interpolation: A Review*. In: MASON, J. C.; COX, M. G. (Hrsg.): *Algorithms for Approximation*. New York, Oxford: Clarendon Press, Oxford University Press, 1987, 143–167. ISBN 0-198-53612-7
- [PP10] PLEVRIS, V.; PAPADRAKAKIS, M.: *A Hybrid Particle Swarm-Gradient Algorithm for Global Structural Optimization*. In: *Computer-Aided Civil and Infrastructure Engineering* 26 (2010), Nr. 1, S. 48–68. ISSN 1093-9687. DOI 10.1111/j.1467-8667.2010.00664.x
- [Puz09] PUZIKOV, A.: *ShaderMan.Next*. <http://www.dream.com.ua/thetool.html>, Abruf: 04.06.2015
- [Pyt16] PYTHON SOFTWARE FOUNDATION: *Python*. <https://www.python.org/>, Abruf: 03.02.2016
- [RB13] RAGHAVAN, B.; BREITKOPF, P.: *Asynchronous evolutionary shape optimization based on high-quality surrogates: application to an air-conditioning duct*. In: *Engineering with Computers* 29 (2013), Nr. 4, S. 467–476. ISSN 0177-0667. DOI 10.1007/s00366-012-0263-0

- [RBN15] RYBERG, A.-B.; BÄCKRYD, R. D.; NILSSON, L.: *A metamodel-based multidisciplinary design optimization process for automotive structures*. In: *Engineering with Computers* 31 (2015), Nr. 4, S. 711–728. ISSN 0177–0667. DOI 10.1007/s00366-014-0381-y
- [RDE⁺15] ROMERO, L.; DOMÍNGUEZ, I. A.; ESPINOSA, M. M.; DOMÍNGUEZ, M.; JIMÉNEZ, M.: *Collaborative Engineering in Distance University Master's Degree*. In: GÓMEZ CHOVA, L.; LÓPEZ MARTÍNEZ, A.; CANDEL TORRES, I. (Hrsg.): *INTED 2015 Conference Proceedings*. Madrid, Spanien, 02.–04.03.2015. Madrid: IATED Academy, ISBN 978–84–606–5763–7
- [Rec73] RECHENBERG, I.: *Evolutionsstrategie: Optimierung technischer Systeme nach Prinzipien der biologischen Evolution*. Bd. 15. Stuttgart-Bad Cannstatt: Frommann-Holzboog, 1973. ISBN 3–7728–0373–3
- [RGGB07] RICK, T.; GROMA, I.; GRÁNICZ, A.; BERCSEY, T.: *Task Scheduling and Resource Allocation with Multi-Variable Heuristics*. In: BOCQUET, J.-C. (Hrsg.): *Proceedings of ICED 07, the 16th International Conference on Engineering Design*. Paris, Frankreich, 28.–31.07.2007, S. 615–616. ISBN 1–904670–02–4
- [RMS98] RAMM, E.; MAUTE, K.; SCHWARZ, S.: *Conceptual design by structural optimization*. In: BORST, R. de; BICANIC, N.; MANG, H.; MESCHKE, G.; BORST, R. d. (Hrsg.): *Computational modelling of concrete structures*. Badgastein, Österreich, 31.03.–03.04.1998. Rotterdam: A.A. Balkema, S. 879–896. ISBN 9–05410–946–7
- [RN04] RAJAN, S. D.; NGUYEN, D. T.: *Design optimization of discrete structural systems using MPI-enabled genetic algorithm*. In: *Structural and Multidisciplinary Optimization* 28 (2004), Nr. 5, S. 340–348. ISSN 1615–147X. DOI 10.1007/s00158-004-0412-1
- [Rob49] ROBINSON, J. B.: *On the Hamiltonian Game (A Traveling-Salesman Problem)*. RAND Research Memorandum RM-303, 1949
- [Roy70] ROYCE, W. W.: *Managing the Development of Large Software Systems*. In: *Proceedings of IEEE Wescon*. August 1970, The Institute of Electrical and Electronics Engineers, S. 328–338
- [RS12] RAO, R. V.; SAVSANI, V. J.: *Mechanical Design Optimization Using Advanced Optimization Techniques*. London: Springer-Verlag, 2012. ISBN 978–1–4471–2747–5
- [SBH10] SIEBERTZ, K.; BEBBER, DAVID THEO VAN; HOCHKIRCHEN, T.: *Statistische Versuchsplanung: Design of Experiments (DoE)*. 1. Aufl. Heidelberg, Dordrecht, London, New York: Springer-Verlag, 2010. ISBN 978–3–64205–492–1
- [SBL98] STEINLE, C.; BRUCH, H.; LAWA, D.: *Projektmanagement: Instrument effizienter Dienstleistung*. 2. Aufl. Frankfurt am Main: Frankfurter Allg. Zeitung, Verlagsbereich Buch, 1998. ISBN 3–929368–27–7

- [SC97] SMITH, A. E.; COIT, D. W.: *Penalty Functions*. In: BÄCK, T.; FOGEL, D. B.; MICHALEWICZ, Z. (Hrsg.): *Handbook of Evolutionary Computation*. Bristol, Philadelphia, New York: Institute of Physics Pub., Oxford University Press, 1997, S. C5.2:1–C5.2:6. ISBN 978-0-75030-895-3
- [Sch84] SCHAFFER, J. D.: *Multi Objective Optimization with Vector Evaluated Genetic Algorithms*. Dissertation. Vanderbilt University, Nashville, TN, 1984
- [Sch85] SCHAFFER, J. D.: *Multiple Objective Optimization with Vector Evaluated Genetic Algorithms*. In: GREFENSTETTE, J. J. (Hrsg.): *Proceedings of the First International Conference on Genetic Algorithms and Their Applications*. Pittsburg, PA, 24.–26.07.1985. Mahwah, NJ: Lawrence Erlbaum Associates, ISBN 0-8058-0426-9
- [Sch86] SCHITTKOWSKI, K.: *NLPQL: A Fortran Subroutine Solving Constrained Nonlinear Programming Problems*. In: *Annals of Operations Research* 5 (1986), Nr. 2, S. 485–500. DOI 10.1007/BF02022087
- [Sch01a] SCHITTKOWSKI, K.: *NLPQLP: A New Fortran Implementation of a Sequential Quadratic Programming Algorithm for Parallel Computing*. Technischer Bericht, Universität Bayreuth, 2001
- [Sch01b] SCHWARZ, S.: *Sensitivitätsanalyse und Optimierung bei nichtlinearem Strukturverhalten*. Dissertation. Universität Stuttgart, Stuttgart, 2001
- [Sch13] SCHUMACHER, A.: *Optimierung mechanischer Strukturen: Grundlagen und industrielle Anwendungen*. 2., aktual. u. erw. Aufl. Berlin, Heidelberg: Springer-Verlag, 2013. ISBN 978-3-64234-699-6
- [SD08] SIVANANDAM, S. N.; DEEPA, S. N.: *Introduction to Genetic Algorithms*. Berlin, Heidelberg, New York: Springer-Verlag, 2008. ISBN 978-3-54073-189-4
- [SD13] SRIVASTAVA, R.; DEB, K.: *An evolutionary based Bayesian design optimization approach under incomplete information*. In: *Engineering Optimization* 45 (2013), Nr. 2, S. 141–165. ISSN 0305-215X. DOI 10.1080/0305215X.2012.661730
- [See05] SEEGER, H.: *Design technischer Produkte, Produktprogramme und -systeme: Industrial Design Engineering*. 2., bearb. u. erw. Aufl. Berlin, Heidelberg: Springer-Verlag, 2005. ISBN 3-540-23653-8
- [Sex15] SEXAUER, J.: *NetworkX Viewer*. https://github.com/jsexauer/networkrx_viewer, Abruf: 04.06.2015
- [SG15] STEINBUCH, R.; GEKELER, S.: *Bionic Optimization in Structural Design: Stochastically Based Methods to Improve the Performance of Parts and Assemblies*. Berlin, Heidelberg: Springer-Verlag, 2015. ISBN 978-3-66246-595-0
- [SH07] SHAH-HOSSEINI, H.: *Problem Solving by Intelligent Water Drops*. In: *IEEE Congress on Evolutionary Computation*. Singapur, 25.–28.09.2007. Piscataway, NJ: IEEE, S. 3226–3231. ISBN 978-1-4244-1339-3. DOI 10.1109/CEC.2007.4424885

- [Sha70] SHANNO, D. F.: *Conditioning of Quasi-Newton Methods for Function Minimization*. In: *Mathematics of Computation* 24 (1970), Nr. 111, S. 647. DOI 10.1090/S0025-5718-1970-0274029-X
- [SHC11] SHELDON, A.; HELWIG, E.; CHO, Y.-B.: *Investigation and Application of Multi-Disciplinary Optimization for Automotive Body-in-White Development*. In: *8th European LS-DYNA Conference*. Strasbourg, Frankreich, 23.–24.05.2011
- [SHH62] SPENDLEY, W.; HEXT, G. R.; HIMSWORTH, F. R.: *Sequential Application of Simplex Designs in Optimisation and Evolutionary Operation*. In: *Technometrics* 4 (1962), Nr. 4, S. 441. ISSN 0040-1706. DOI 10.2307/1266283
- [Sie15] SIEMENS PLM: *NX 10 Hilfe*. Siemens Product Lifecycle Management Software Inc., 2015
- [SK00] SOBIESKI, I. P.; KROO, I. M.: *Collaborative Optimization Using Response Surface Estimation*. In: *AIAA Journal* 38 (2000), Nr. 10, S. 1931–1938. ISSN 0001-1452. DOI 10.2514/2.847
- [SL14] SAUTHOFF, B.; LACHMAYER, R.: *Generative Design Approach for Modelling of Large Design Spaces*. In: BRUNOE, T. D.; NIELSEN, K.; JOERGENSEN, K. A.; TAPS, S. B. (Hrsg.): *Proceedings of the 7th World Conference on Mass Customization, Personalization, and Co-Creation (MCPC 2014)*. Aalborg, Dänemark, 04.–07.02.2014. Cham: Springer-Verlag, 2014, S. 241–251. ISBN 978-3-319-04270-1. DOI 10.1007/978-3-319-04271-8_21
- [SLC01] SIMPSON, T. W.; LIN, D.; CHEN W.: *Sampling Strategies for Computer Experiments: Design and Analysis*. In: *International Journal of Reliability and Applications* 2 (2001), Nr. 3, S. 209–240
- [SM63] SCHMIT, LUCIEN A. JR.; MALLETT, R. H.: *Structural Synthesis and Design Parameter Hierarchy*. In: *Journal of the Structural Division* 89 (1963), Nr. 4, S. 269–300
- [SM13] SIGMUND, O.; MAUTE, K.: *Topology optimization approaches: A comparative review*. In: *Structural and Multidisciplinary Optimization* 48 (2013), Nr. 6, S. 1031–1055. ISSN 1615-147X. DOI 10.1007/s00158-013-0978-6
- [Sof15] SOFTWARE- UND ORGANISATIONS-SERVICE: *JobScheduler*. <https://www.sos-berlin.com/jobscheduler>, Abruf: 21.12.2015
- [SPKA01] SIMPSON, T. W.; POPLINSKI, J. D.; KOCH, P. N.; ALLEN, J. K.: *Metamodels for Computer-based Engineering Design: Survey and recommendations*. In: *Engineering with Computers* 17 (2001), Nr. 2, S. 129–150. ISSN 0177-0667. DOI 10.1007/PL00007198
- [SPS+06] SMITH, A. R.; PRYER, K. M.; SCHUETTPPELZ, E.; KORALL, P.; SCHNEIDER, H.; WOLF, P. G.: *A Classification for Extant Ferns*. In: *Taxon* 55 (2006), Nr. 3, S. 705–731. ISSN 0040-0262. DOI 10.2307/25065646
- [SR08] SEIFFERT, U.; RAINER, G. (Hrsg.): *Virtuelle Produktentstehung für Fahrzeug und Antrieb im Kfz: Prozesse, Komponenten, Beispiele aus der Praxis*. Wiesbaden: Vieweg + Teubner Verlag, 2008. ISBN 978-3-8348-0345-0

- [SS89] SOBIESZCZANSKI-SOBIESKI, J.: *Optimization by Decomposition: A Step from Hierarchic to Non-Hierarchic Systems*. In: BARTHELEMY, J.-F. M. (Hrsg.): *Recent Advances in Multidisciplinary Analysis and Optimization: NASA Conference Publication 3031 Part 1*. Hampton, VA, USA, 28.–30.09.1988. Washington: NASA Langley Research Center, 1989, S. 58–78
- [SSAPS03] SOBIESZCZANSKI-SOBIESKI, J.; ALTUS, T. D.; PHILLIPS, M.; SANDUSKY, R.: *Bilevel Integrated System Synthesis for Concurrent and Distributed Processing*. In: *AIAA Journal* 41 (2003), Nr. 10, S. 1996–2003. ISSN 0001–1452. DOI 10.2514/2.1889
- [SSAS00] SOBIESZCZANSKI-SOBIESKI, J.; AGTE, J. S.; SANDUSKY, R. R.: *Bilevel Integrated System Synthesis*. In: *AIAA Journal* 38 (2000), Nr. 1, S. 164–172. ISSN 0001–1452. DOI 10.2514/2.937
- [SSS11] SANCHEZ, E.; SCHILLACI, M.; SQUILLERO, G.: *Evolutionary Optimization: the muGP toolkit*. New York, Dordrecht, Heidelberg, London: Springer-Verlag, 2011. ISBN 978–0–38709–425–0
- [SST15] SCHMELCHER, J.; STETTER, R.; TILL, M.: *Integrating the Ability for Topology Optimization in a Commercial CAD-System*. In: WEBER, C.; HUSUNG, S.; CASCINI, G.; CANTAMESSA, M.; MARJANOVIĆ, D. (Hrsg.): *Proceedings of the 20th International Conference on Engineering Design (ICED 15), Vol. 6: Design Methods and Tools - Part 2*. Mailand, Italien, 27.–30.07.2015, S. 173–182. ISBN 978–1–90467–069–8
- [SVM96] SCHLIERKAMP-VOOSEN, D.; MÜHLENBEIN, H.: *Adaptation of Population Sizes by Competing Subpopulations*. In: *IEEE International Conference on Evolutionary Computation*. Nagoya, Japan, 20.–22.05.1996, IEEE, S. 330–335. ISBN 0–7803–2902–3. DOI 10.1109/ICEC.1996.542384
- [SW15] STANGL, T.; WARTZACK, S.: *Feature Based Interpretation and Reconstruction of Structural Topology Optimization Results*. In: WEBER, C.; HUSUNG, S.; CASCINI, G.; CANTAMESSA, M.; MARJANOVIĆ, D. (Hrsg.): *Proceedings of the 20th International Conference on Engineering Design (ICED 15), Vol. 6: Design Methods and Tools - Part 2*. Mailand, Italien, 27.–30.07.2015, S. 235–244. ISBN 978–1–90467–069–8
- [SWMW89] SACKS, J.; WELCH, W. J.; MITCHELL, T. J.; WYNN, H. P.: *Design and Analysis of Computer Experiments*. In: *Statistical Science* 4 (1989), Nr. 4, S. 409–423. DOI 10.1214/ss/1177012413
- [SYZ12] SHI, L.; YANG, R. J.; ZHU, P.: *A method for selecting surrogate models in crashworthiness optimization*. In: *Structural and Multidisciplinary Optimization* 46 (2012), Nr. 2, S. 159–170. ISSN 1615–147X. DOI 10.1007/s00158-012-0760-1
- [TA10] THONEMANN, U.; ALBERS, M.: *Operations-Management: Konzepte, Methoden und Anwendungen*. 2., aktualisierte Aufl. München: Pearson Studium, 2010. ISBN 978–3–8632–6559–5
- [Tec10] TECKLENBURG, G.: *Die parametrisch assoziative Konstruktion im Entwicklungsprozess Karosserie: Einführung und Ziele*. In: TECKLENBURG, G. (Hrsg.): *Die digitale Produktentwicklung*. Renningen: Expert-Verlag, 2010, S. 1–8. ISBN 978–3–8169–2961–1

- [Tex15] TEXTOR, J.: *Welcome to DAGitty*. <http://www.dagitty.net/>, Abruf: 04.06.2015
- [The15] THE GNOME PROJECT: *Dia*. <https://wiki.gnome.org/Apps/Dia/>, Abruf: 04.06.2015
- [THM02] TOPCUOGLU, H.; HARIRI, S.; MIN-YOU WU: *Performance-Effective and Low-Complexity Task Scheduling for Heterogeneous Computing*. In: *IEEE Transactions on Parallel and Distributed Systems* 13 (2002), Nr. 3, S. 260–274. DOI 10.1109/71.993206
- [TN16] TOEPFER, F.; NAUMANN, T.: *Management of Vehicle Architecture Parameters*. In: MARJANOVIĆ, D.; ŠTORGA, M.; PAVKOVIĆ, N.; BOJČETIĆ, N.; ŠKEC, S. (Hrsg.): *Proceedings of the 14th International Design Conference (DESIGN 2016)*. Cavtat, Dubrovnik, Kroatien, 16.–19.05.2016, S. 1679–1688
- [Toa15] TOAL, D. J. J.: *A study into the potential of GPUs for the efficient construction and evaluation of Kriging models*. In: *Engineering with Computers* 32 (2015), Nr. 3, S. 377–404. ISSN 0177–0667. DOI 10.1007/s00366-015-0421-2
- [TTL05] THAIN, D.; TANNENBAUM, T.; LIVNY, M.: *Distributed Computing in Practice: The Condor Experience*. In: *Concurrency and Computation: Practice and Experience* 17 (2005), Nr. 2-4, S. 323–356. DOI 10.1002/cpe.938
- [TV02] TZANNETAKIS, N.; VAN DE PEER, J.: *Design optimization through parallel-generated surrogate models, optimization methodologies and the utility of legacy simulation software*. In: *Structural and Multidisciplinary Optimization* 23 (2002), S. 170–186. ISSN 1615–147X
- [UIN13] UCHIDA, A.; ITO, Y.; NAKANO, K.: *Accelerating ant colony optimisation for the travelling salesman problem on the GPU*. In: *International Journal of Parallel, Emergent and Distributed Systems* 29 (2013), Nr. 4, S. 401–420. ISSN 1744–5760. DOI 10.1080/17445760.2013.842568
- [Ulb06] ULBRICH, M.: *Methoden der mathematischen Optimierung*. In: GRABE, J. (Hrsg.): *Optimierung in der Geotechnik*. Hamburg, 10.10.2006, Technische Universität Hamburg-Harburg, Institut für Geotechnik und Baubetrieb, ISBN 3–93631–013–0
- [Uni15] UNIVERSITY OF LIVERPOOL: *High Throughput Computing using Condor: Desktop Condor (DTCondor) User Guide*. <http://condor.liv.ac.uk/DTCondor/>, Abruf: 21.12.2015
- [Vaj14] VAJNA, S. (Hrsg.): *Integrated Design Engineering: Ein interdisziplinäres Modell für die ganzheitliche Produktentwicklung*. Berlin, Heidelberg: Springer, 2014. ISBN 978–3–64241–103–8
- [Vaj15] VAJNA, S.: *Attributes in Integrated Design Engineering - A new Way to Describe Both Performance Capability and Behaviour of a Product*. In: WEBER, C.; HUSUNG, S.; CASCINI, G.; CANTAMESSA, M.; MARJANOVIĆ, D. (Hrsg.): *Proceedings of the 20th International Conference on Engineering Design (ICED 15), Vol. 2: Design Theory and Research Methodology Design*. Mailand, Italien, 27.–30.07.2015, S. 127–136. ISBN 978–1–90467–065–0

- [Van84] VANDERPLAATS, G. N.: *An Efficient Feasible Directions Algorithm for Design Synthesis*. In: *AIAA Journal* 22 (1984), Nr. 11, S. 1633–1640. ISSN 0001–1452. DOI 10.2514/3.8829
- [VCJB05] VAJNA, S.; CLEMENT, S.; JORDAN, A.; BERCSEY, T.: *The Autogenetic Design Theory: An Evolutionary View of the Design Process*. In: *Journal of Engineering Design* 16 (2005), Nr. 4, S. 423–440. ISSN 0954–4828. DOI 10.1080/09544820500267781
- [VDI93] VDI-RICHTLINIE 2221: *Methodik zum Entwickeln und Konstruieren technischer Systeme und Produkte*. Beuth Verlag, 1993
- [VENKJ07] VAJNA, S.; EDELMANN-NUSSER, J.; KITTEL, K.; JORDAN, A.: *Optimisation of a bow riser using the autogenetic design theory*. In: *Journal of Engineering Design* 18 (2007), Nr. 5, S. 525–540. ISSN 0954–4828. DOI 10.1080/09544820701403839
- [Vin15] VINTHER, M.: *Diagram Designer 1.28*. <http://logicnet.dk/DiagramDesigner/>, Abruf: 04.06.2015
- [VJK13] VAJNA, S.; JORDAN, A.; KITTEL, K.: *Weiterentwicklung der Autogenetischen Konstruktionstheorie*. Abschlussbericht zum Forschungsvorhaben VA 134/10-2, 2013
- [VK10] VAJNA, S.; KITTEL, K.: *Weiterentwicklung der Autogenetischen Konstruktionstheorie*. Abschlussbericht zum Forschungsvorhaben VA 134/10-1, 2010
- [VKB11] VAJNA, S.; KITTEL, K.; BERCSEY, T.: *The Autogenetic Design Theory: Product Development as an Analogy to Biological Evolution*. In: BIRKHOFER, H. (Hrsg.): *The Future of Design Methodology*. London: Springer-Verlag, 2011, S. 169–179. ISBN 978–0–85729–614–6. DOI 10.1007/978-0-85729-615-3_15
- [VMC95] VOIGT, H.-M.; MÜHLENBEIN, H.; CVETKOVIC, D.: *Fuzzy Recombination for the Continuous Breeder Genetic Algorithm*. In: ESHELMAN, L. J. (Hrsg.): *Proceedings of the Sixth International Conference on Genetic Algorithms*. Pittsburgh, PA, 15.–19.07.1995. San Francisco: Kaufman, ISBN 1–558–60370–0
- [VWBZ09] VAJNA, S.; WEBER, C.; BLEY, H.; ZEMAN, K.: *CAX für Ingenieure: Eine praxisbezogene Einführung*. 2., völlig neu bearb. Aufl. Berlin, Heidelberg: Springer, 2009. ISBN 978–3–540–36038–4
- [WCF+14] WOOD, M.; CHEN, P.; FU, K.; CAGAN, J.; KOTOVSKY, K.: *The Role of Design Team Interaction Structure on Individual and Shared Mental Models*. In: GERO, J. S. (Hrsg.): *Design Computing and Cognition '12*. Dordrecht: Springer Science+Business Media, 2014, S. 209–226. ISBN 978–94–017–9111–3. DOI 10.1007/978-94-017-9112-0_12
- [WD16] WÜNSCH, A.; DZIUBACZYK, B.: *Condor GUI*. <http://sourceforge.net/projects/condor-gui/?source=directory>, Abruf: 27.01.2016

- [Web05] WEBER, C.: *CPM/PDD - An Extended Theoretical Approach to Modeling Products and Product Development Processes*. In: BLEY, H.; JANSEN, H.; KRAUSE, F.-L.; SHPITALNI, M. (Hrsg.): *Advances in Methods and Systems for the Development of Products and Processes: Proceedings of the 2. German-Israeli Symposium on Design and Manufacture*. Berlin, 07.–08.07.2005. Stuttgart: Fraunhofer IRB, S. 159–179. ISBN 3–8167–6874–1
- [Wei15] WEICKER, K.: *Evolutionare Algorithmen*. 3. Aufl. Wiesbaden: Springer Vieweg, 2015. ISBN 978–3–658–09957–2
- [WF12] WACHOWIAK, M. P.; FOSTER LAMBE, A. E.: *GPU-Based Asynchronous Global Optimization with Particle Swarm*. In: *Journal of Physics: Conference Series* 385 (2012). DOI 10.1088/1742-6596/385/1/012012
- [WJV15] WÜNSCH, A.; JORDAN, A.; VAJNA, S.: *Simultaneous Optimisation: Strategies for Using Parallelization Efficiently*. In: WEBER, C.; HUSUNG, S.; CASCINI, G.; CANTAMESSA, M.; MARJANOVIĆ, D. (Hrsg.): *Proceedings of the 20th International Conference on Engineering Design (ICED 15), Vol. 6: Design Methods and Tools - Part 2*. Mailand, Italien, 27.–30.07.2015, S. 133–142. ISBN 978–1–90467–069–8
- [WLSL06] WANG, C.; LIN, Y.; SOHRABY, K.; LI, B.: *An adaptive algorithm for active queue management*. In: *Journal of Combinatorial Optimization* 12 (2006), Nr. 1-2, S. 151–162. DOI 10.1007/s10878-006-8909-8
- [WPBS88] WINNER, R. I.; PENNELL, J. P.; BERTRAND, H. E.; SLUSAREZUK, M. M. G.: *The Role of Concurrent Engineering in Weapons System Acquisition*. IDA-Report R-338, Institute for Defense Analyses, 1988
- [WPF05] WIECZOREK, M.; PRODAN, R.; FAHRINGER, T.: *Scheduling of Scientific Workflows in the ASKALON Grid Environment*. In: *ACM SIGMOD Record* 34 (2005), Nr. 3, S. 56. DOI 10.1145/1084805.1084816
- [WPV16] WÜNSCH, A.; PILZ, F.; VAJNA, S.: *Morphix: An Evolutionary Way to Support Conceptual Design*. In: MARJANOVIĆ, D.; ŠTORGA, M.; PAVKOVIĆ, N.; BOJČETIĆ, N.; ŠKEC, S. (Hrsg.): *Proceedings of the 14th International Design Conference (DESIGN 2016)*. Cavtat, Dubrovnik, Kroatien, 16.–19.05.2016, S. 769–778
- [WRBB96] WUJEK, B. A.; RENAUD, J. E.; BATILL, S. M.; BROCKMAN, J. B.: *Concurrent Subspace Optimization Using Design Variable Sharing in a Distributed Computing Environment*. In: *Concurrent Engineering* 4 (1996), Nr. 4, S. 361–377
- [WTVS13] WÜNSCH, A.; TSAY, D.-M.; VAJNA, S.; SCHABACKER, M.: *A New Approach for Designing a Gearbox for a New Kind of Independently Controllable Transmissions*. In: *ASME 2013 International Design Engineering Technical Conferences and Computers and Information in Engineering Conference*. Portland, OR, 04.–07.08.2013. New York: American Society of Mechanical Engineers, ISBN 978–0–79185–588–1. DOI 10.1115/DETC2013-12952
- [Wün15] WÜNSCH, A.: *NX 10 für Fortgeschrittene - kurz und bündig*. 1. Aufl. Wiesbaden: Springer Vieweg, 2015. ISBN 978–3–658–09188–0

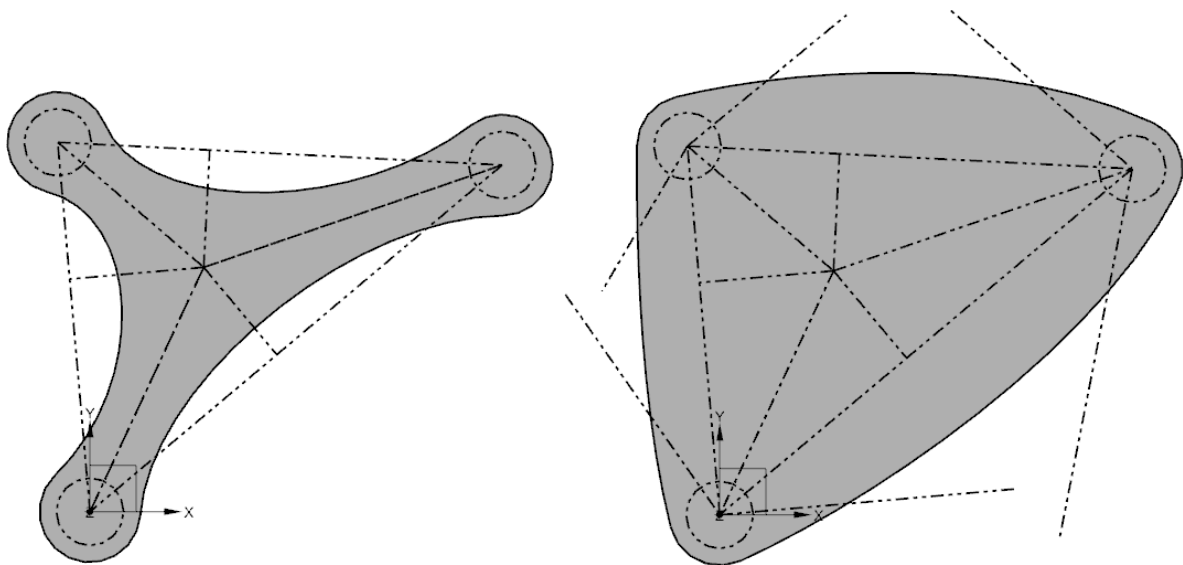
- [WV15] WÜNSCH, A.; VAJNA, S.: *Effiziente Parallelisierung bei Optimierungsproblemen in der Produktentwicklung*. In: KASPER, R.; GROTE, K.-H. (Hrsg.): *12. Magdeburger Maschinenbautage*. Magdeburg, 30.09.–01.10.2015, ISBN 978–3–94472–226–9
- [YA96] YU-KWONG KWOK; AHMAD, I.: *Dynamic Critical-Path Scheduling: An Effective Technique for Allocating Task Graphs to Multiprocessors*. In: *IEEE Transactions on Parallel and Distributed Systems* 7 (1996), Nr. 5, S. 506–521. DOI 10.1109/71.503776
- [YPY10] YEH, T.-M.; PAI, F.-Y.; YANG, C.-C.: *Performance improvement in new product development with effective tools and techniques adoption for high-tech industries*. In: *Quality & Quantity* 44 (2010), Nr. 1, S. 131–152. DOI 10.1007/s11135-008-9186-7
- [yWo15] YWORKS: *yEd Graph Editor*. <http://www.yworks.com/products/yed>, Abruf: 04.06.2015
- [YWT⁺05] YANG, R. J.; WANG, N.; THO, C. H.; BOBINEAU, J. P.; WANG, B. P.: *Metamodeling Development for Vehicle Frontal Impact Simulation*. In: *Journal of Mechanical Design* 127 (2005), Nr. 5, S. 1014. DOI 10.1115/1.1906264
- [Zbi12] ZBIERSKI, M.: *A Simulated Annealing Algorithm for GPU Clusters*. In: WYRZYKOWSKI, R.; DONGARRA, J.; KARCZEWSKI, K.; WAŚNIEWSKI, J. (Hrsg.): *Parallel Processing and Applied Mathematics*. Berlin, Heidelberg: Springer-Verlag, S. 750–759. ISBN 978–3–642–31463–6
- [ZDT00] ZITZLER, E.; DEB, K.; THIELE, L.: *Comparison of Multiobjective Evolutionary Algorithms: Empirical Results*. In: *Evolutionary Computation* 8 (2000), Nr. 2, S. 173–195. ISSN 1063–6560. DOI 10.1162/106365600568202
- [Zit99] ZITZLER, E.: *Evolutionary Algorithms for Multiobjective Optimization: Methods and Applications*. Dissertation. Institut für Technische Informatik und Kommunikationsnetze, Eidgenössische Technische Hochschule Zürich, Zürich, CH, 1999
- [ZLT01] ZITZLER, E.; LAUMANN, M.; THIELE, L.: *SPEA2: Improving the Strength Pareto Evolutionary Algorithm*. TIK-Report 103, Computer Engineering and Networks Laboratory (TIK), Swiss Federal Institute of Technology (ETH) Zurich, 2001
- [Zom96] ZOMAYA, A. Y.: *Parallel and Distributed Computing Handbook*. New York: McGraw-Hill, 1996. ISBN 978–0–07073–020–5
- [Zou60] ZOUTENDIJK, G.: *Methods of Feasible Directions: A Study in Linear and Non-Linear Programming*. Elsevier, 1960
- [ZP13] ZEGARD, T.; PAULINO, G. H.: *Toward GPU accelerated topology optimization on unstructured meshes*. In: *Structural and Multidisciplinary Optimization* 48 (2013), Nr. 3, S. 473–485. ISSN 1615–147X. DOI 10.1007/s00158-013-0920-y
- [ZP16] ZEGARD, T.; PAULINO, G. H.: *Bridging topology optimization and additive manufacturing*. In: *Structural and Multidisciplinary Optimization* 53 (2016), Nr. 1, S. 175–192. ISSN 1615–147X. DOI 10.1007/s00158-015-1274-4

- [ZW10] ZHANG, L.; WIRTH, A.: *On-line machine scheduling with batch setups*. In: *Journal of Combinatorial Optimization* 20 (2010), Nr. 3, S. 285–306. ISSN 1382–6905. DOI 10.1007/s10878-009-9211-3
- [Zwi66] ZWICKY, F.: *Entdecken, Erfinden, Forschen im morphologischen Weltbild*. München, Zürich: Droemer/Knaur, 1966
- [ZZF14] ZHU, B.; ZHANG, X.; FATIKOW, S.: *A multi-objective method of hinge-free compliant mechanism optimization*. In: *Structural and Multidisciplinary Optimization* 49 (2014), Nr. 3, S. 431–440. ISSN 1615–147X. DOI 10.1007/s00158-013-1003-9

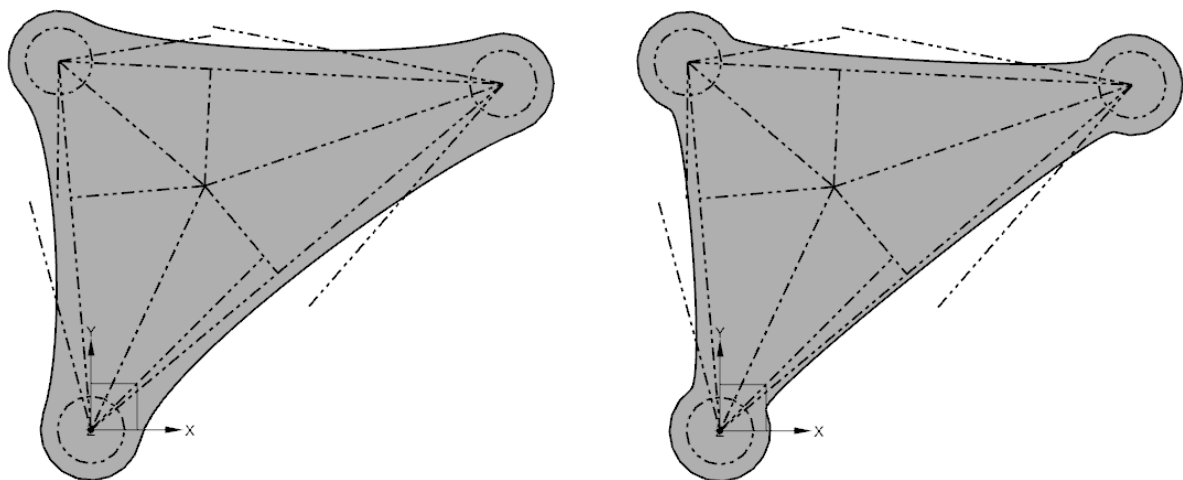
Anhang

A.1 Parametrische CAD-Modellierung

Abbildung A.1 zeigt die Flexibilität eines parametrisierten CAD-Modells unter Verwendung einer Steuerskizze und NURBS. In den oberen beiden Abbildungen wird der Winkel der Linien variiert, auf denen die Kontrollpunkte positioniert sind. In den unteren beiden Abbildungen wird die Position der Kontrollpunkte auf diesen Linien verändert. Die Parametrisierung der Skizzen ist in Abbildung 2.14 dargestellt.



Variation der Winkel der die Kontrollpunkte positionierenden Linien



Variation der Kontrollpunktposition auf den Linien bei konstanten Winkeln

Abbildung A.1: Flexibilität eines parametrisierten CAD-Modells unter Verwendung einer Steuerskizze und NURBS

A.2 Evolutionäre Algorithmen

A.2.1 Binäre Codierung von Designvariablen

Die folgende Tabelle zeigt den Unterschied zwischen der Binärcodierung und der Cray-Codierung bei der Codierung einer ganzen Zahl unter der Verwendung von vier Bits. Bei der Cray-Codierung wird durch die Invertierung eines Bits die codierte Zahl um 1 erhöht oder reduziert.

Tabelle A.1: Vergleich von Binärcodierung und Cray-Codierung, nach [Gol89]

Ganze Zahl	Binär	Cray
0	0 0 0 0	0 0 0 0
1	0 0 0 1	0 0 0 1
2	0 0 1 0	0 0 1 1
3	0 0 1 1	0 0 1 0
4	0 1 0 0	0 1 1 0
5	0 1 0 1	0 1 1 1
6	0 1 1 0	0 1 0 1
7	0 1 1 1	0 1 0 0
8	1 0 0 0	1 1 0 0
9	1 0 0 1	1 1 0 1
10	1 0 1 0	1 1 1 1
11	1 0 1 1	1 1 1 0
12	1 1 0 0	1 0 1 0
13	1 1 0 1	1 0 1 1
14	1 1 1 0	1 0 0 1
15	1 1 1 1	1 0 0 0

A.2.2 Beispiele evolutionärer Algorithmen

Tabelle A.2 gibt eine Übersicht bekannter und weitverbreiteter evolutionärer und genetischer Algorithmen mit besonderem Fokus auf multikriteriellen Anwendungen. Ein Vergleich der Effektivität der aufgeführten Algorithmen unter Verwendung verschiedener multikriterieller Testprobleme wird in [ZDT00] gegeben. Eine Gegenüberstellung multikriterieller evolutionärer Algorithmen ist in [Kun05] zu finden.

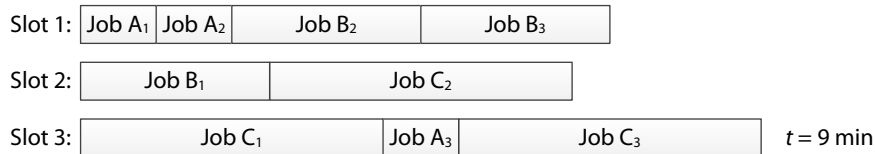
Tabelle A.2: Übersicht evolutionärer und genetischer Algorithmen, nach [SBH10]

Kurzform	Bezeichnung	Quellen
VEGA	Vector Evaluated Genetic Algorithms	[Sch84], [Sch85]
HLGA	Hajela & Lin Genetic Algorithm	[HL92]
MOGA	Multiple Objective Genetic Algorithm	[FF93], [FF98]
NPGA	Niched Pareto Genetic Algorithm	[HNG94]
SPEA2	Strength Pareto Evolutionary Algorithm	[Zit99], [ZLT01] [HNG94]
NSGA-II	Nondominated Sorting Genetic Algorithm	[DPAM00], [DPAM02], [Kan08], [Ins12]
PAES	Pareto Archived Evolution Strategy	[KC00], [Ins12]
PESA	Pareto Envelope-Based Selection Algorithm	[CKO00]
ϵ -MOEA	Steady State Multi-objective Evolutionary Algorithm	[DMM03]

A.3 Scheduling von Berechnungsjobs

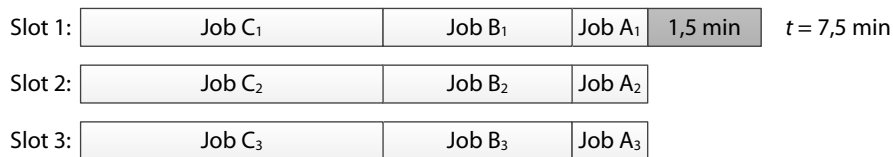
A.3.1 Prioritätenbasierte Bearbeitung

First In - First Out:



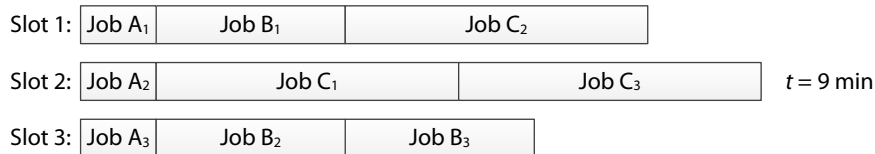
Prioritätenbasiert:

$\Delta t = 1,5 \text{ min} \approx 16,7 \%$



(a) Keine Abhängigkeit und keine Restriktion

First In - First Out:



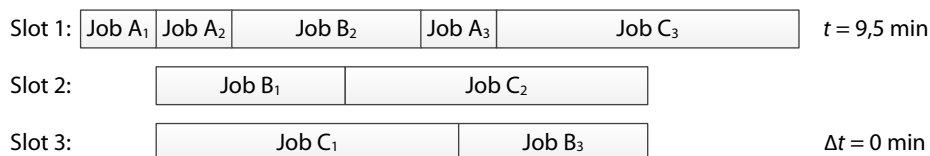
$\Delta t = 1,5 \text{ min} \approx 16,7 \%$

Prioritätenbasiert:

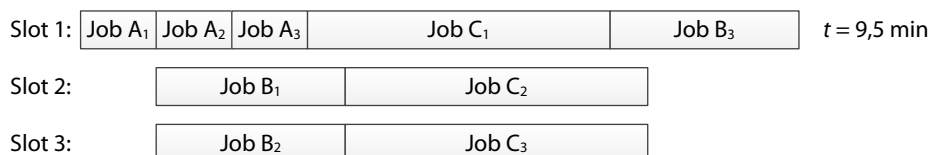


(b) Abhängigkeit A - BC und keine Restriktion

First In - First Out:



Prioritätenbasiert:



(c) Abhängigkeit A - BC und Restriktion 1 x A

Abbildung A.2: Reduzierung der Laufzeit durch prioritätenbasierte Bearbeitung bei der Verwendung von drei Berechnungsslots

A.3.2 Scheduling-Verfahren

Tabelle A.3: Verfahren für das Scheduling von Berechnungsjobs, nach [Cöl16]

ID	Domäne	Ressourcen	Zielsetzung	Alternativen	Ergebnis	Quelle
1	Prozesse eines Projektplans	Humane und physische Ressourcen	Minimierung der Entwicklungszeit	GA, Heuristik kritischer Pfad	GA, Heuristik kritischer Pfad	[RGGB07]
2	Komplexe DAG	Homogene CPU	Minimierung der Schedulingzeit	Hybride PSO-GA, Metaheuristik, Optimierungsalgorithmen (Sim. Annealing, GA, HLFET)	Hybride PSO-GA, Metaheuristik	[KV16]
3	DAG	Heterogene CPU	Minimierung der Laufzeit	Critical-Path-on-a-Processor (CPOP), Heterogeneous Earliest-Finish-Time (HEFT)	HEFT-Algorithmus	[THM02]
4	Workflow	CPU	Minimierung der Laufzeit, Minimierung der Schedulingzeit	HEFT, GA, Myopic-Algorithmus	HEFT-Algorithmus	[WPF05]
5	Datenpakete (Buffer)	Empfänger innerhalb eines Netzwerkes	Kontrolle von Datenverlust und Queue-Länge zur Vermeidung von Überlastung und Datenverlust	Proportional Integral (PI), Adaptive Proportional Integral (API)	Adaptive Proportional Integral (API)	[WLSL06]
6	Jobs mit Abhängigkeiten	CPU	Minimierung der Laufzeit	Online Scheduling (dyn. Allokation) durch Approximationsalgorithmus	Großer Einfluss von Abhängigkeiten, CPU-Anzahl und Netzwerkstruktur	[FKST98]

Tabelle A.3: Verfahren für das Scheduling von Berechnungsjobs (Fortsetzung)

ID	Domäne	Ressourcen	Zielsetzung	Alternativen	Lösung	Quelle
7	Jobs in einem optischen Netzwerk	Parallele Rechner inkl. CPU, RAM Netzwerk	Maximierung des Durchsatzes aller Jobs	Spezialisierte Approximationsalgorithmen	Durchsatzsteigerung um 20 %	[CSX05]
8	Jobs mit Abschlusszeiten	Homogene parallele Rechner	Maximierung der Jobs, die ihre Deadline einhalten, Minimierung der Zeit der Ressourcenzuweisung (Allokation)	Pseudopolynomialzeitalgorithmen	Pseudopolynomialzeitalgorithmen	[LSSY10]
9	Zwei Gruppen unterschiedlicher Jobs	Ein Rechner	Minimierung der Laufzeit einer Gruppe während Laufzeit der anderen konstant bleibt	Spezialisierte Polynomialzeitalgorithmen und Pseudopolynomialzeitalgorithmen, versch. Ziel-funktionen	Pseudopolynomialzeitalgorithmen	[LCF15]
10	Mehrere Jobs werden in Batches zusammengefasst	Parallele Rechner	Minimierung der Laufzeit, Festlegung von Grenzen	Heuristic Uniform-Batch-Size (UBS), Heuristic Wait-Half-Setup (WHS), Heuristic Sync (Sy)	Spezielle Heuristiken	[ZW10]
11	DAG (Aufgaben-graph)	CPU	Minimierung der Laufzeit	Dyn. kritischer Pfad (statische Allokation), versch. Algorithmen	Dyn. kritischer Pfad (statische Allokation)	[YA96]

A.4 Cluster-Management-Systeme

Tabelle A.4: Anforderungsliste zur Recherche und Auswahl eines Cluster-Management-Systems [Dzi15]

OvGU Magdeburg LMI		Anforderungsliste <i>Cluster-Management-System</i>	Blatt 1 Seite 1
Änd.	F/W	Anforderungen	Verantwortl.
		<u>Plattform:</u>	
	F	Kompatibel mit Windows 7 Professional oder höher	
	W	Kompatibel mit Windows Server Editionen sowie anderen Betriebssystemen	
		<u>Lizenzierung:</u>	
	F	Frei zugänglich (Freeware)	
	W	Offener Quellcode (Open source)	
		<u>Benutzungsfreundlichkeit:</u>	
		<i>a) Installation/Konfiguration:</i>	
	W	Assistent	
	W	Möglichst keine lokalen Anpassungen	
		<i>b) Nutzung:</i>	
	F	Kommandozeile	
	W	Grafische Oberfläche	
		<i>c) Dokumentation:</i>	
	F	Benutzerhandbuch	
	W	Mailing Listen/Support	
		<u>Rechnermanagement:</u>	
	W	Heterogenität der Ressourcen (bspw. Prozessorarchitektur)	
	F	Gruppierung von Maschinen	
	F	Ranking von Maschinen	
	W	Verteilte Besitzrechte	
		<u>Jobmanagement:</u>	
	F	Erstellung von Wiederherstellungspunkten	
	F	Priorisierung von Jobs	
	F	Pausierung von Jobs	

Tabelle A.4: Anforderungsliste zur Recherche und Auswahl eines Cluster-Management-Systems (Fortsetzung)

OvGU Magdeburg LMI		Anforderungsliste <i>Cluster-Management-System</i>	Blatt 1 Seite 1
Änd.	F/W	Anforderungen	Verantwortl.
	F	Wiederaufnahme von Jobs	
	F	Abhängigkeiten von Jobs	
	F	Keine Beeinflussung auf gesperrtem Rechner durch User	
	W	Parallele Ausführung von Jobs auf einer Maschine	
		<u>Ressourcenmanagement:</u>	
	F	Bestimmung von benötigter Software	
	F	Begrenzung von Rechnerressourcen (bspw. CPUs)	
	F	Festlegen von Mindestanforderungen an einen Rechner	
		<u>Energiemanagement:</u>	
	W	Ruhezustand/Standby	
	W	Wake On LAN (WOL)	
		<u>Lizenzmanagement:</u>	
	W	Konfiguration von unterschiedlichen Lizenztypen	
		<u>Sicherheit:</u>	
	F	Nutzer-Authentifizierung	
	W	Zugriffsbeschränkungen	
		<u>Monitoring:</u>	
	F	Überwachung des Clusters (GUI)	
	F	Auswertung des Clusters in Form von Statistiken	
	F	Fehlerrückmeldung und -auswertung (z. B. Log-Datei)	
		<u>Zusätzliche Funktionen:</u>	
	F	Programmierschnittstellen (Java, Python, C/C++ etc.)	
	W	Virtualisierung	
	W	Cloudnutzung	

Tabelle A.5: Windows-basierte Cluster-Management-Systeme, nach [WJV15], [Dzi15]

ID	Name	Plattform	Lizenz	Vorteile	Nachteile	Quelle
1	DTCCondor	Windows	Free-ware	GUI, wenig Programmierung, Monitoring, heterogene Cluster, Kenntnis von HTCCondor nicht notwendig	Optimiert für die Anwendung an der Universität Liverpool	[Uni15]
2	HTCCondor	Linux, Mac OS X, Windows	Open source	Eigenschaften eines klassischen Batch-Systems, flexibles Ressourcen-Management, heterogene Cluster, vielseitiger Einsatz, vollständige Dokumentation und Quellcode	Keine GUI, Monitoring durch externe Software	[Cen15]
3	Job Scheduler	Linux, Windows, Cloud	Open source	Dt. Dokumentation, Web-basierte GUI (XML) und Konsole, Backup-Cluster, API zu Java, Perl, VBScript	MySQL Datenbank und http Server benötigt, Komplexe Anleitung	[Sof15]
4	JPPF	Java-Environment	Open source	Sicheres Netzwerk, Cloud-Nutzung vorbereitet, durchdachtes Management und Monitoring, FTP, keine zusätzliche Konfiguration notwendig	Eigene API, kaum Details zu Lizenz und Komplexität	[Coh15]

Tabelle A.5: Windows-basierte Cluster-Management-Systeme (Fortsetzung)

ID	Name	Plattform	Lizenz	Vorteile	Nachteile	Quelle
5	Oddjob	Linux, Windows	Open source	Visuelle Konfiguration, Monitoring und Steuerung von Jobs über das Netzwerk, XML, Konfigurationsda- teien, umfangreiche Dokumentation	Java und Datenbank benötigt	[Gor14]
6	Open Grid Scheduler	Linux, Mac OS X, Windows	Open source	Konsole, Monitoring, API zu C/C++, Java, Perl, Python	Komplexe Anleitung, wenig Doku- mentation, Kompilierung notwendig, keine GUI	[Gri13]
7	GNUbatch	Linux, Windows	Open source	GUI und Kommandozeile, komplexe und verknüpfte Jobs, Reportfunktion, gute Dokumentation	Komplexes User-Interface, Kompilierung notwendig, Java und C Kenntnisse	[Fre09]
8	schedulix	Linux, Windows	Open source	Dt. Dokumentation, Web Interface zur Modellierung, Monitoring u. Verwaltung, einfache Benutzerführung, Kommandozeile für API (Java, Python, Perl)	Wenig Doku- mentation, hoher adminis- trativer Aufwand	[Ind15]

Tabelle A.6: Binärer Vergleich der Funktionalität ausgewählter Cluster-Management-Systeme [Dzi15]

	GNUbatch	JobSched.	JPPF	Oddjob	HTCondor	OGS
GNUbatch		1	1	1	1	0
JobScheduler	0		1	0	1	0
JPPF	0	0		0	1	0
Oddjob	0	1	1		1	0
HTCondor	0	0	0	0		0
OGS	1	1	1	1	1	
Summe	1	3	4	2	5	0
Rangfolge	5	3	2	4	1	6

Tabelle A.7: Binärer Vergleich der Benutzbarkeit ausgewählter Cluster-Management-Systeme [Dzi15]

	GNUbatch	JobSched.	JPPF	Oddjob	HTCondor	OGS
GNUbatch		1	0	1	1	0
JobSched.	0		0	1	1	0
JPPF	1	1		1	1	1
Oddjob	0	0	0		0	0
HTCondor	0	0	0	1		0
OGS	1	1	0	1	1	
Summe	2	3	0	5	4	1
Rangfolge	4	3	6	1	2	5

Tabelle A.8: Binärer Vergleich der Umsetzbarkeit ausgewählter Cluster-Management-Systeme [Dzi15]

	GNUbatch	JobSched.	JPPF	Oddjob	HTCondor	OGS
GNUbatch		0	1	0	1	0
JobSched.	1		1	1	1	0
JPPF	0	0		0	0	0
Oddjob	1	0	1		1	0
HTCondor	0	0	1	0		0
OGS	1	1	1	1	1	
Summe	3	1	5	2	4	0
Rangfolge	3	5	1	4	2	6

A.5 Grafische Modellierungssoftware

Tabelle A.9: Windows-basierte grafische Modellierungssoftware

ID	Name	Relevanz	Vorteile	Nachteile	Quelle
1	Depends	1	Grafische Grapherstellung, Open Source, Python	–	[Lin14]
2	yEd Graph Editor	2	Grafische Grapherstellung, Export in verschiedene Dateiformate möglich, Freeware	Kein Open Source	[yWo15]
3	Dia	2	Python API, Export möglich	Veraltete GUI, viele Anpassungen notwendig	[The15]
4	PyQtGraph	2	Umfangreiche Einsatzmöglichkeiten	Komplette Eigenentwicklung notwendig	[Cam12]
5	PyQt Diagram Editor	2	Grafische Grapherstellung	Veraltete GUI, viele Anpassungen notwendig	[Bou15]
6	Flowchart Custom Node	2	–	Veraltete GUI, zu viele Anpassungen notwendig	[Kun15]
7	Graphviz	3	–	Keine intuitive Benutzung	[EGH ⁺ 15]
8	Calligra Flow	3	Grafische Grapherstellung, Open Source	Zu großer Funktionsumfang	[Cal15]
9	QNodes-Editor	3	Grafische Grapherstellung	C++, zu viele Anpassungen notwendig	[ALG12]
10	ShaderMan. Next	3	OpenGL, Python	zu viele Anpassungen notwendig	[Puz09]
11	DAG program	4	–	Definition nur über Tabelle	[Knü11]
12	JSPlumb Toolkit	4	–	Rein webbasiert, kostenpflichtig	[JSP15]

Tabelle A.9: Windows-basierte grafische Modellierungssoftware (Fortsetzung)

ID	Name	Relevanz	Vorteile	Nachteile	Quelle
13	NetworkX	4	Grapherstellung, Python	Komplette Eigenentwicklung notwendig, keine interaktive Erstellung	[Net14]
14	Networkx Viewer	4	Python	Reine Visualisierung	[Sex15]
15	Diagram Designer	5	–	Kein direkter Export in Quellcode möglich	[Vin15]
16	DAGitty	5	–	Webbasiert, kein direkter Export in Quellcode möglich	[Tex15]
17	TETRAD	5	–	Nur binäres Dateiformat	[GSSR15]

A.6 Optimierung der Prozessprioritäten in DAG2OPT

A.6.1 Overhead-Zeit von HTCCondor

Abbildung A.3 zeigt die Ergebnisse der experimentellen Ermittlung der Overhead-Zeiten von HTCCondor. Die Overhead-Zeit beträgt mindestens 7s. Aufgrund von unregelmäßig auftretendem externen Datenverkehr im Netzwerk kommt es zu Schwankungen in den Overhead-Zeiten. Der Mittelwert der Overhead-Zeit beträgt 10s bei einer Standardabweichung von 4s.

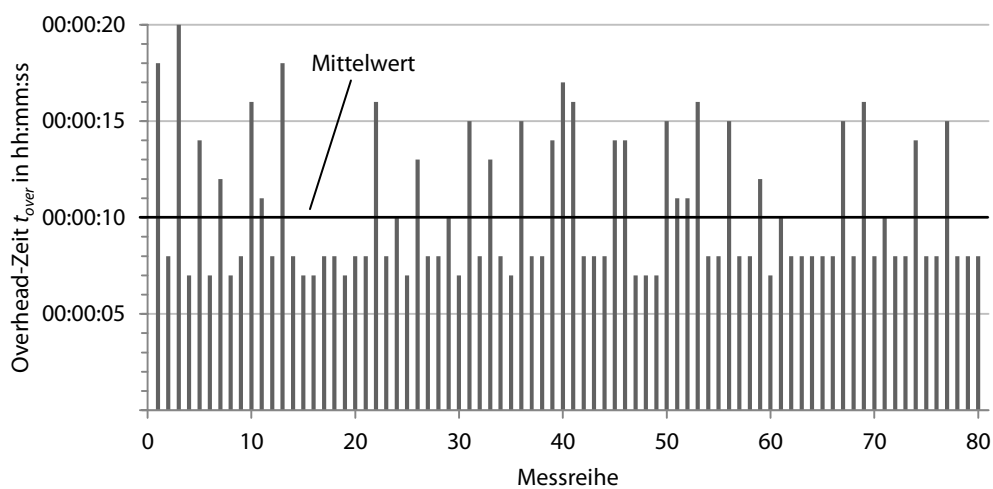


Abbildung A.3: Experimentelle Ermittlung der durchschnittlichen Overhead-Zeit von HTCCondor, nach [Cöl16]

A.6.2 Allokationsverhalten von HTCCondor

Das Allokationsverhalten von HTCCondor kann auf der Basis von verschiedenen Versuchen durch die folgenden Regeln beschrieben werden [Cöl16]:

1. Jobs mit einer höheren Priorität werden in der Queue Jobs mit einer niedrigeren Priorität vorgezogen.
2. Haben in der Queue mehrere Jobs die gleiche Priorität, erfolgt die Bearbeitung nach dem FIFO-Prinzip.
3. In einem DAG muss ein Job (Child) stets die gleiche oder eine höhere Priorität wie seine direkten Vorgänger (Parents) besitzen. Ist dies nicht der Fall, wird die Priorität des Jobs überschrieben. Existieren mehrere direkte Vorgänger, ist der höchste Prioritätswert maßgebend.
4. Ein Job eines DAG wird erst in die Queue geladen, wenn die Bearbeitung seiner direkten Vorgänger abgeschlossen ist. Dieser Ladevorgang dauert mehrere Sekunden und kann in seiner Dauer variieren.
5. Die Overhead-Zeit zwischen zwei Jobs, welche nacheinander auf einem Slot bearbeitet werden, beträgt durchschnittlich 10 s (s. Abbildung A.3).

Abbildung A.4 zeigt die Überschreibung der Prioritäten eines DAG durch HTCCondor (Regel 3). Somit wird stets die vertikale Fertigstellung eines DAG sichergestellt [Cen15].

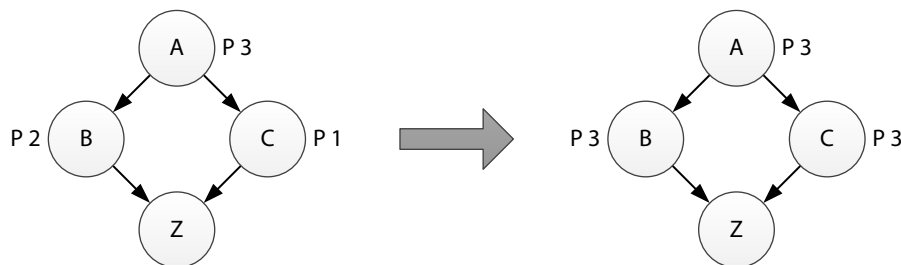


Abbildung A.4: Überschreibung der Prozessprioritäten eines DAG durch HTCCondor, nach [Cöl16]

A.6.3 Simulation der Allokation von HTCCondor

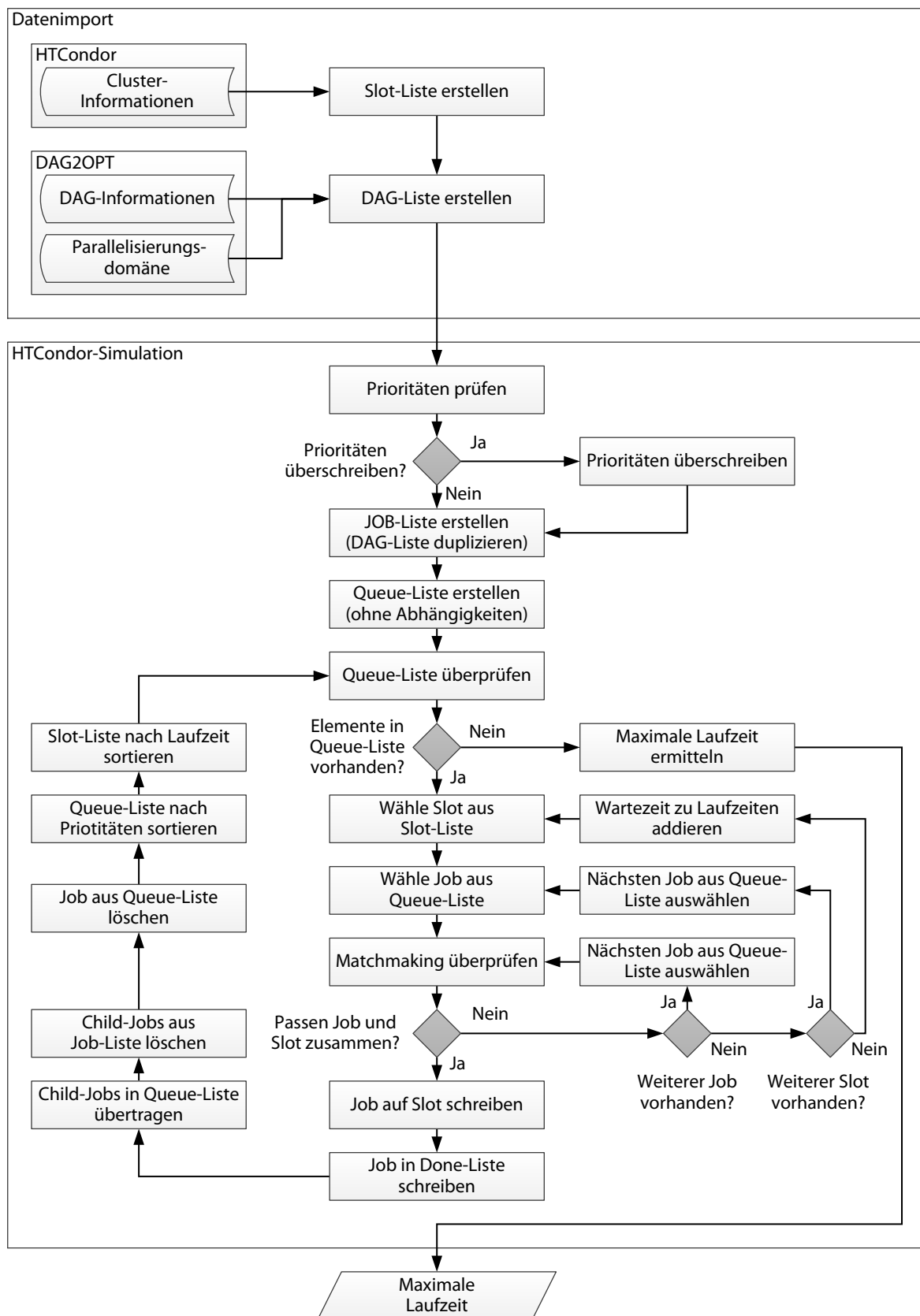


Abbildung A.5: Aktivitätsdiagramm der Allokationssimulation von HTCCondor, nach [Cö116]

A.6.4 Anteil optimaler Prioritätswerte für das Testszenario 2

Abbildung A.6 zeigt die auf Basis der vollständigen Enumeration ermittelten prozentualen Anteile optimaler Prioritätswerte an den gesamten möglichen Kombinationen von Prioritätswerten für das Testszenario 2. Nimmt die Größe der Parallelisierungsdomäne die Werte $n_p = 4; 7; 15; 22; 23; 28; 31; 36; 38; 39$ an, führen lediglich 4,7% der möglichen Kombinationen von Prioritätswerten zur minimalen Laufzeit.

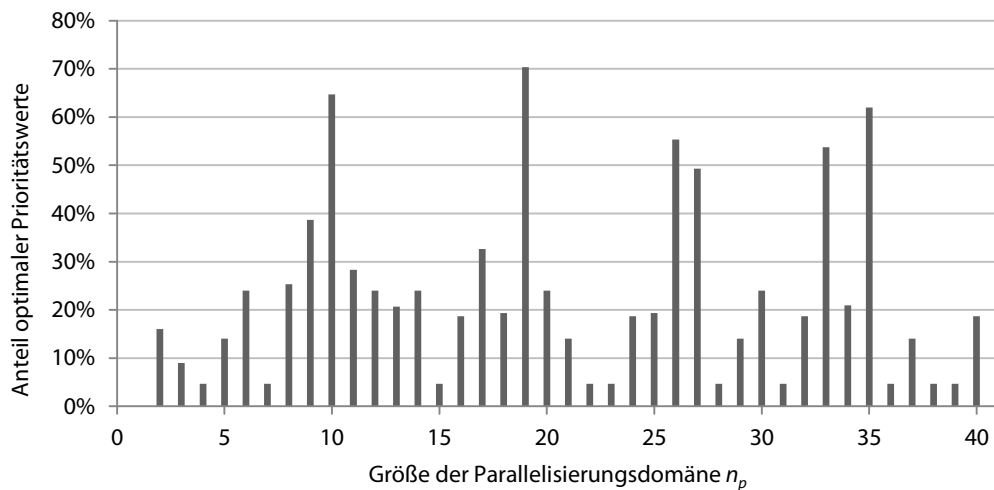


Abbildung A.6: Prozentualer Anteil optimaler Prioritätswerte für das Testszenario 2, nach [Cöl16]

A.6.5 Testscenario zur Überprüfung der Funktionsweise

Testscenario 3 stellt das komplexeste der drei Testscenarien dar. Wie in Abbildung A.7 dargestellt, besteht der DAG aus acht Prozesselementen, deren Laufzeiten im Gegensatz zu den ersten beiden Testscenarien deutlich länger sind.

Zudem werden in den Ressourcen Software- und Lizenzrestriktionen berücksichtigt. Zur Berücksichtigung dieser Restriktionen bei der Allokation ist jedem Prozesselement die für die Bearbeitung benötigte Software zugeordnet. Dabei werden das CAx-System Siemens NX und das FE-System CalculiX (CX) [DW16] verwendet.

Der Cluster in diesem Testscenario besteht aus 4 Slots bzw. Rechnern ($n_s = 4$), wobei NX auf allen Slots und CX lediglich auf 3 von 4 Slots vorhanden ist. Die parallele Verwendung von NX ist auf 2 Lizenzen beschränkt, CX besitzt keine Lizenzbeschränkung.

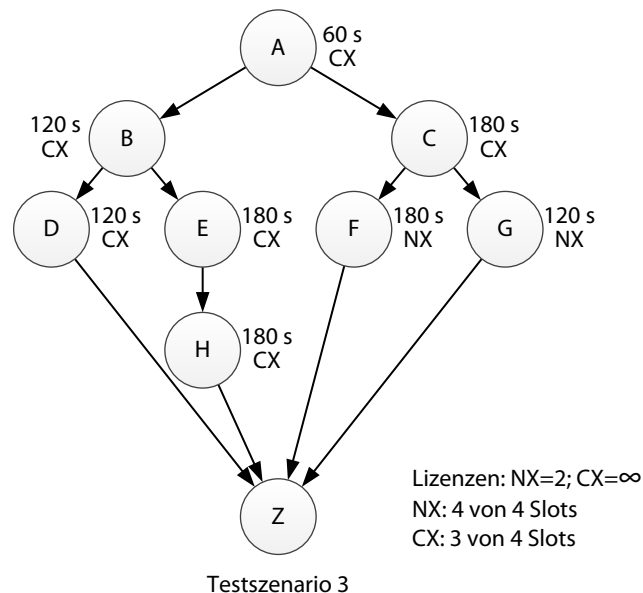


Abbildung A.7: Testscenario 3 zur Überprüfung der Funktionsweise des Werkzeugs zur Optimierung der Prozessprioritäten ($n_s = 4$), nach [Cöl16]

A.6.6 Laufzeitapproximation der Optimierung

Die Laufzeitvorhersage der Optimierung der Prozessprioritäten im Vorfeld der Optimierung erfolgt auf Basis der Laufzeit der Allokationssimulation von HTCCondor und der prognostizierten Anzahl an Evaluationen der Optimierung. Dazu wird zunächst die Laufzeit der Allokationssimulation in Abhängigkeit von den Eingabeparametern n_p , n_s und n_e untersucht. Die Eingabeparameter werden gemäß den in Tabelle A.10 aufgelisteten Werten miteinander kombiniert, wodurch sich 210 mögliche Wertekombinationen ergeben. Dieses Vorgehen entspricht somit einem vollfaktoriellen Versuchsplan, in dem alle möglichen Parameterkombinationen überprüft werden (s. Abschnitt 2.7.1).

Tabelle A.10: Designvariablen und Werte der Laufzeitapproximation

Designvariable	Werte
n_p – Größe der Parallelisierungsdomäne	2, 5, 10, 20, 50, 100, 150
n_s – Anzahl der Slots zur parallelen Verarbeitung	2, 5, 10, 20, 50, 100
n_e – Anzahl der Prozesselemente eines DAG	2, 4, 6, 8, 10

Die Wertebereiche der Eingabeparameter stellen repräsentative Bereiche dar, in denen bei einer Optimierung in der Regel die Auswahl der Werte erfolgt. Die Eingangsparameter n_p und n_s sind bei niedrigen Werten feiner aufgelöst, um in diesen Wertebereichen mehr Informationen über das Laufzeitverhalten zu generieren. Die Variation von n_e erfolgt gleichverteilt. Die aus den aufgelisteten Werten abgeleiteten Kombinationen der Eingabeparameter für die Laufzeituntersuchung der Allokationssimulation ist in Abbildung A.8 dargestellt.

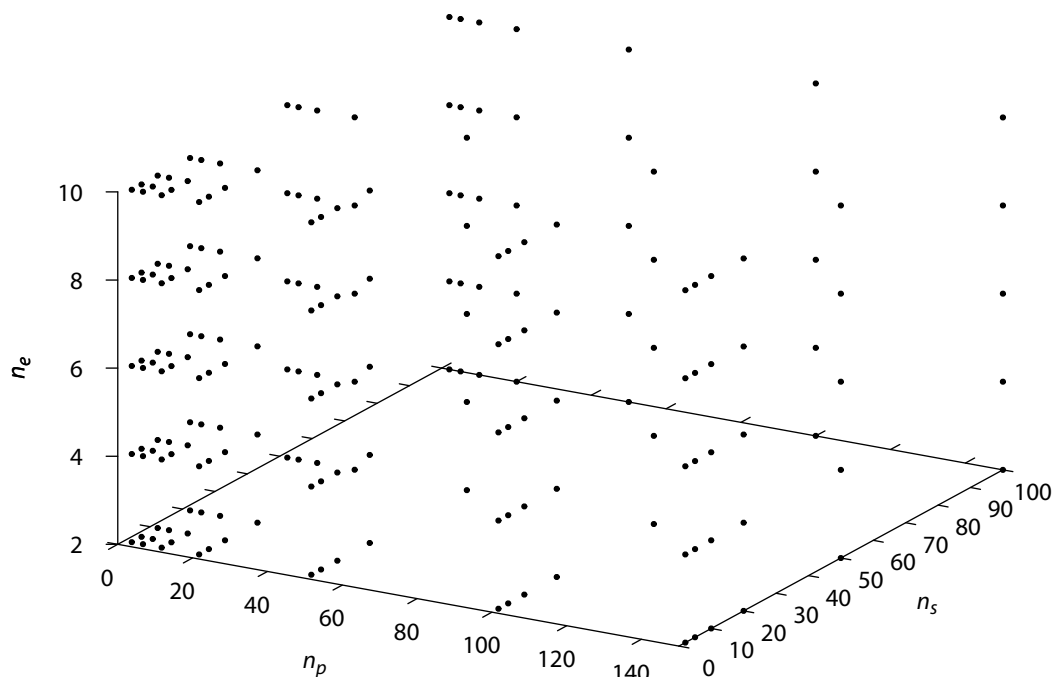


Abbildung A.8: Untersuchte Kombinationen der Eingabeparameter zur Laufzeituntersuchung der Allokationssimulation

Abbildung A.9 zeigt die Laufzeit der Allokationssimulation t_{sim} in Abhängigkeit von der Größe der Parallelisierungsdomäne n_p und der Anzahl der Prozesselemente n_e für definierte Werte der Slotanzahl n_s . Die Knotenpunkte der dargestellten Flächen repräsentieren jeweils die berechneten Stützstellen. Jede Stützstelle stellt den Mittelwert der Laufzeit aus drei Durchläufen der Allokationssimulation dar. Zur Ermittlung der Laufzeiten wurde eine Dell T1600 Workstation (CPU: Intel Xeon E3-1290 3,60 GHz) verwendet.

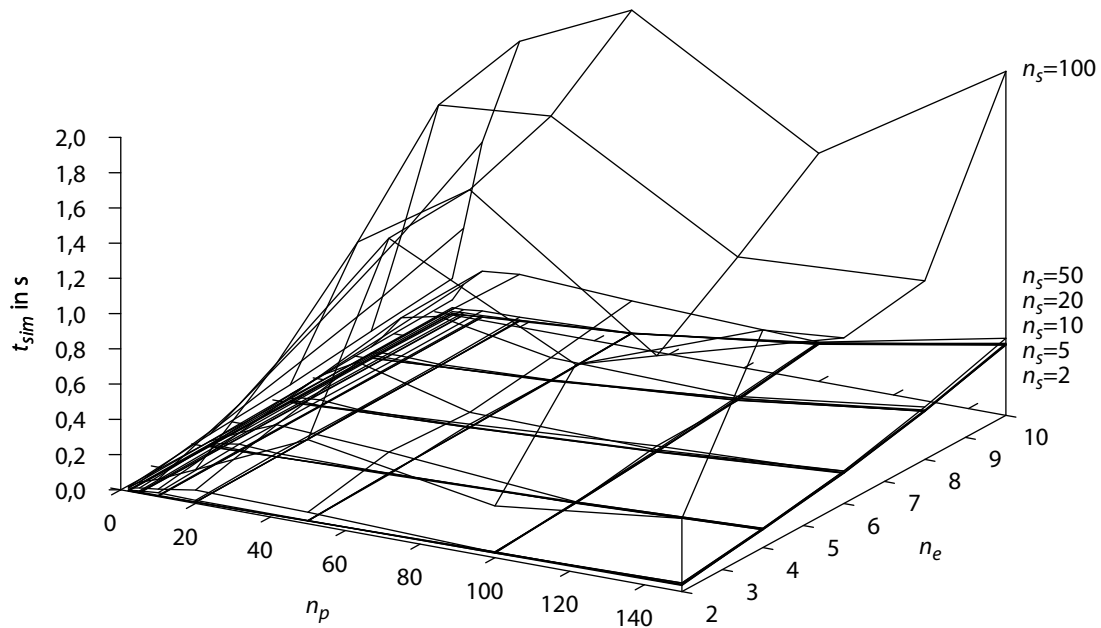


Abbildung A.9: Laufzeit der Allokationssimulation für definierte Werte der Slotanzahl

Die gleiche Abhängigkeit der Laufzeit der Allokationssimulation t_{sim} von den Eingabeparametern ist in Abbildung A.10 zu sehen. Zur Erhöhung der Übersichtlichkeit werden hierbei nur Werte der Slotanzahl $n_s \leq 20$ dargestellt.

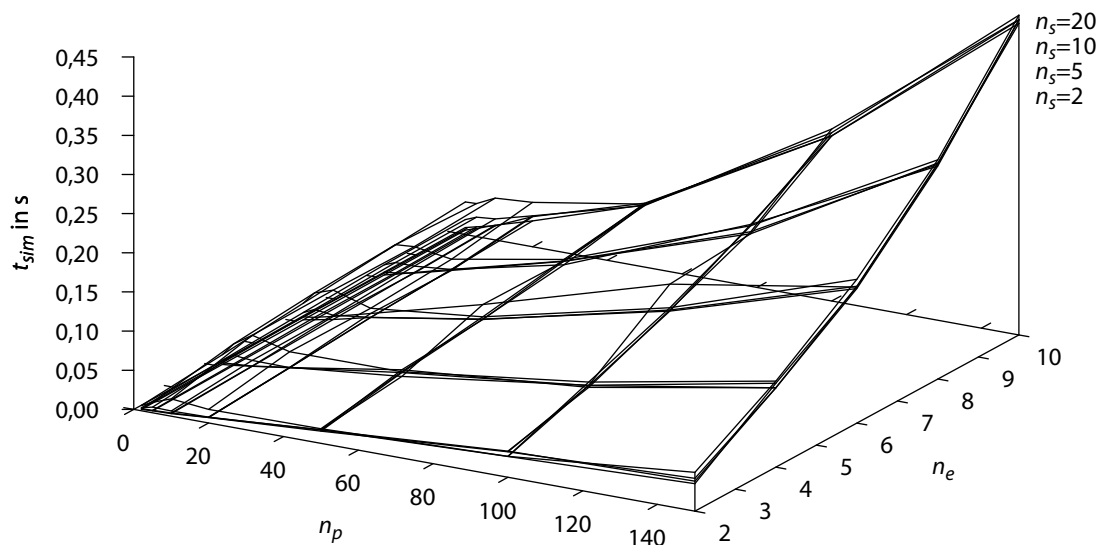


Abbildung A.10: Laufzeit der Allokationssimulation für reduzierte Werte der Slotanzahl

Aus den Abbildungen ist zu erkennen, dass die Laufzeit mit den Eingabeparametern ansteigt, was mit der Anzahl der Iterationen bei der Allokationssimulation zu erklären ist (s. Abbildung A.5). Da in dem Algorithmus zur Allokationssimulation über alle Elemente der Eingabeparameter iteriert wird, sind bei einer größeren Anzahl der Elemente auch mehr Iterationen notwendig. Entspricht die Anzahl der Slots der Größe der Parallelisierungsdomäne ($n_s = n_p$), wird bei jeder Iteration über einen Slot (äußere Iterationsschleife) direkt ein Prozesselement der Queue zugewiesen und es entfallen überflüssige Iterationen der inneren Iterationsschleife, wodurch sich die Laufzeit stark verkürzt.

Auch bei der in Abbildung A.11 dargestellten Abhängigkeit der Laufzeit der Allokationssimulation t_{sim} von der Größe der Parallelisierungsdomäne n_p und der Anzahl der Slots n_s für definierte Werte der Prozesselementanzahl n_e ist das beschriebene Verhalten zu erkennen. Mit zunehmender Anzahl der Prozesselemente n_e steigt die Laufzeit t_{sim} annähernd linear an.

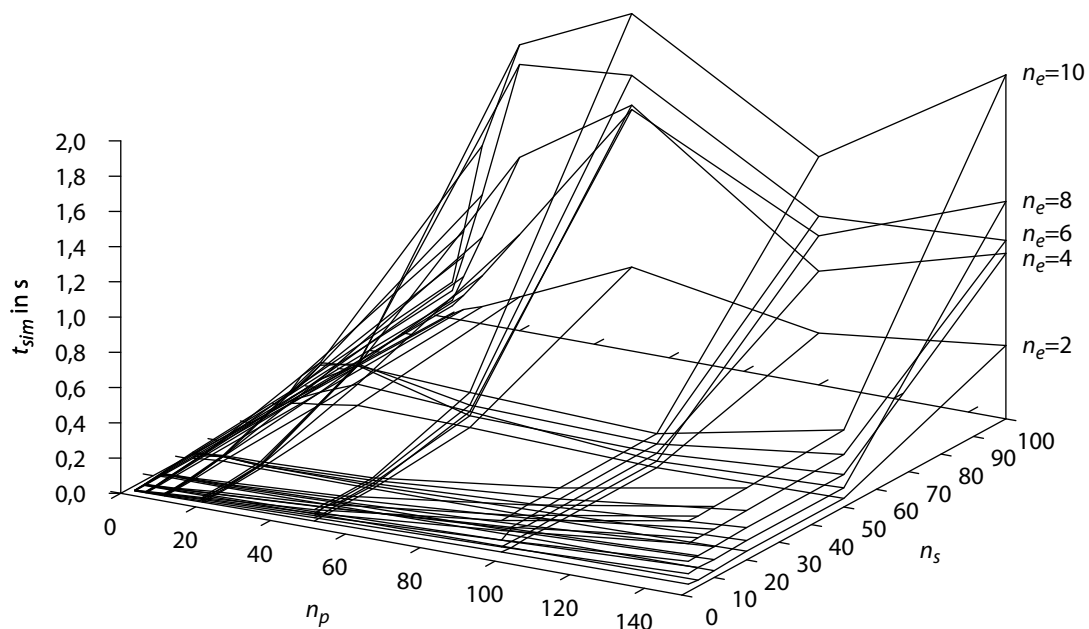


Abbildung A.11: Laufzeit der Allokationssimulation für definierte Werte der Prozesselementanzahl

Die Abhängigkeit der Laufzeit von der Anzahl der Slots n_s und der Anzahl der Prozesselemente n_e ist für definierte Größen der Parallelisierungsdomäne n_p in Abbildung A.12 dargestellt.

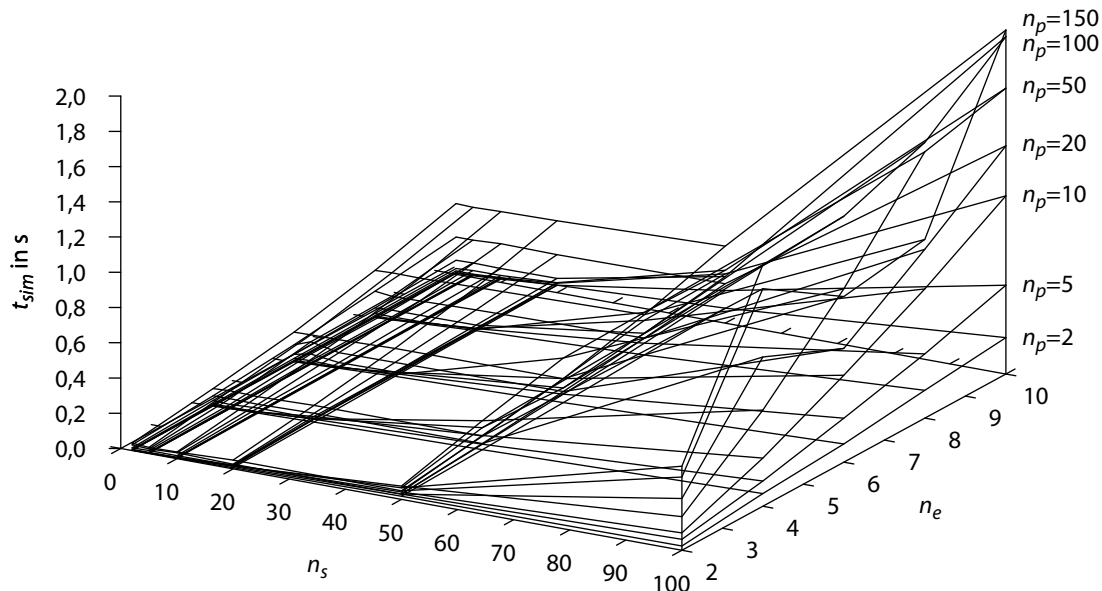


Abbildung A.12: Laufzeit der Allokationssimulation für definierte Größen der Parallelisierungsdomäne

In den Darstellungen der Abhängigkeit der Laufzeit der Allokationssimulation von den Eingabeparametern ist zu erkennen, dass zwischen den Eingabeparametern Wechselwirkungen existieren, wodurch die Laufzeit der Simulation beeinflusst wird. Insbesondere wenn die Anzahl der Slots und die Größe der Parallelisierungsdomäne den gleichen Wert aufweisen ($n_s = n_p$), reduziert sich die Laufzeit der Simulation sehr stark. Da diese Zusammenhänge nur durch ein komplexes Metamodell (s. Abschnitt 2.7.2) abgebildet werden können, was für die reine Vorhersage der Laufzeit der Optimierung der Prozessprioritäten zu aufwendig wäre, wird die Verwendung eines Metamodells nicht weiter verfolgt.

Die Simulation der Allokation von HTCCondor wird sehr schnell durchgeführt. Die maximale durch die Versuche ermittelte Laufzeit beträgt 1,95 s. Da diese Laufzeit vom Anwender als tolerierbar eingeschätzt wird und im Vorfeld der Optimierung eine Allokationssimulation zur Analyse der aktuell verwendeten Prioritätswerte notwendig ist, kann die Laufzeit dieser Simulation zur Abschätzung der gesamten Laufzeit der Optimierung mit der Anzahl der prognostizierten Evaluationen multipliziert werden. Diese einfache Berechnung der Laufzeit ist zulässig, da die Optimierung der Prozessprioritäten aufgrund der kurzen Evaluationszeit rein sequentiell erfolgt. Neben der Laufzeit der Allokationssimulation ist lediglich die Anzahl der Evaluationen maßgebend, welche im Folgenden ermittelt wird.

Zur Gewährleistung einer hohen Wahrscheinlichkeit, das globale Optimum zu finden, ist das Abbruchkriterium des verwendeten NSGA-II Algorithmus indirekt und variabel zur Problemkomplexität formuliert. Die Optimierung läuft solange, bis über 10 Generationen keine Verbesserung des Fitnesswertes erreicht wurde. Hierbei wird die zufällig erzeugte Startpopulation nicht gezählt, wodurch sich bei der Optimierung mindestens 12 Generationen ergeben. Weiterhin ist die Populationsgröße mit dem fünffachen der Anzahl der Prozesselemente ($n_I = 5 \cdot n_e$) ebenfalls variabel zur Problemkomplexität definiert (s. Abschnitt 5.4.2).

Die daraus resultierenden bei einer Optimierung mindestens benötigten Evaluationen sind in Abhängigkeit von der Anzahl der Prozesselemente n_e in Abbildung A.13 dargestellt.

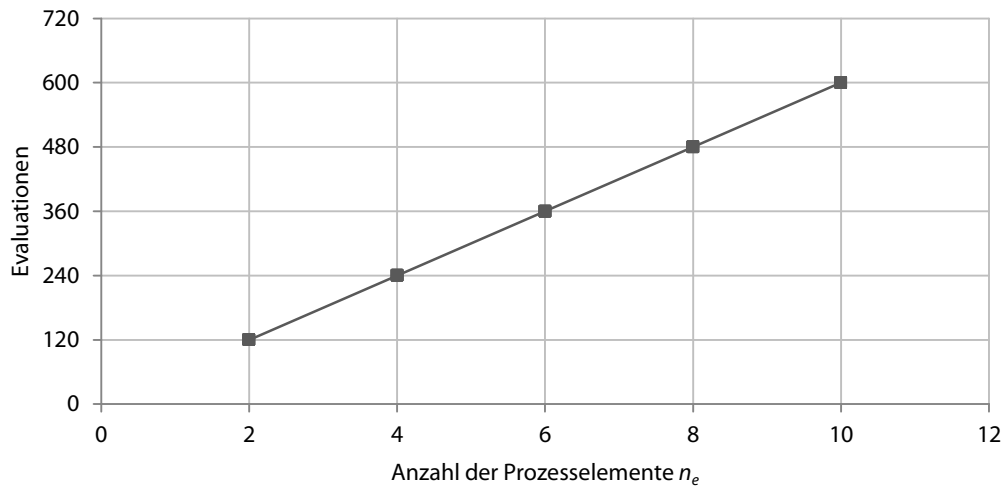


Abbildung A.13: Mindestens benötigte Evaluationen bei der Optimierung der Prozessprioritäten

Die mindestens benötigten Evaluationen stellen somit stets das zwölfwache der Populationsgröße dar, welche sich aus der fünffachen Anzahl der Prozesselemente berechnet ($12 \cdot 5 \cdot n_e$). Aufgrund des Zufallsanteils des verwendeten NSGA-II Algorithmus ist eine präzise Vorhersage der benötigten Evaluationen jedoch nicht möglich. Die in diesem Abschnitt beschriebenen Untersuchungen zur Bestimmung der Laufzeiten erfordern in der Regel lediglich die mindestens benötigten Evaluationen.

Die Vorhersage der zur Optimierung der Prozessprioritäten benötigten Laufzeit erfolgt auf Basis einer im Vorfeld der Optimierung durchgeführten Simulation der Allokation von HTCondor, welche das Evaluationsmodell der Optimierung darstellt. Die Laufzeit der Allokationssimulation wird anschließend mit der voraussichtlichen Anzahl an Evaluationen multipliziert. Da die Laufzeit der Simulation von der Taktrate der CPU des verwendeten Rechners abhängt, wird diese somit bei der Laufzeitvorhersage direkt berücksichtigt.

A.7 Fallstudien

A.7.1 Laufzeitverteilung der Prozesselemente in Fallstudie 3

Der Evaluationsprozess in der Fallstudie 3 besteht aus drei Prozesselementen: der Modellgenerierung und Preprocessing, einem Lastfall für die lineare Statik und einem Lastfall für die Berechnung der Eigenfrequenzen. Die Modellgenerierung und das Preprocessing (A) erfolgt im CAx-System NX 10. Dabei wird das parametrische CAD-Modell nach dem Einlesen der Designvariablenwerte aktualisiert, die Masse des Modells extrahiert, eine step-Datei zur Archivierung exportiert und die Eingabedateien für die Berechnung der Lastfälle generiert. Anschließend werden die beiden Lastfälle mittels des FE-Solvers NX Nastran analysiert (B, C) und der Zielfunktionswert berechnet (Z). Der DAG des Evaluationsprozesses mit den jeweiligen gemittelten Laufzeiten der Prozesselemente ist in Abbildung A.14 dargestellt. Die mittlere Gesamtlaufzeit des dargestellten Evaluationsprozesses beträgt $t = 00:14:15$.

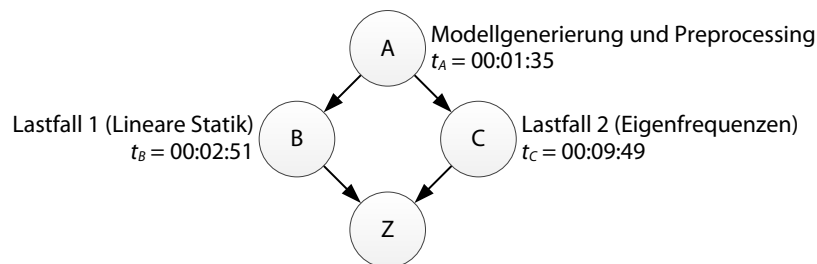


Abbildung A.14: DAG des Evaluationsprozesses der Fallstudie 3

Da die Laufzeiten der Prozesselemente A, B und C einer Streuung unterliegen, werden die Laufzeiten über den Mittelwert aus 230 zufällig generierten Varianten berechnet. Die Laufzeiten der Prozesselemente dienen zur Berechnung der bei der Parallelisierung erreichten Speedup S_p und Effizienz E_p .

Die Laufzeitverteilung des Prozesselementes A ist in Abbildung A.15 dargestellt. Der Mittelwert beträgt 1 min 35 s bei einer Standardabweichung von 56 s.

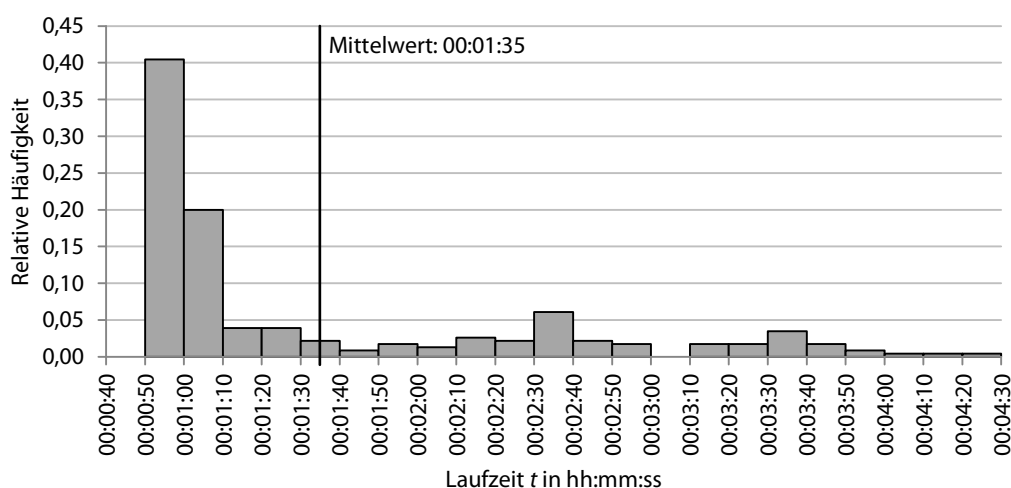


Abbildung A.15: Laufzeitverteilung des Prozesselementes A

Neben der unterschiedlichen Modellgröße, welche zu einer Streuung der Vernetzungszeit führt, ist der Export der step-Datei verantwortlich für die sehr starke Streuung der Laufzeit des Prozesselementes A. Die Laufzeiten streuen weniger stark, wenn keine step-Datei exportiert wird.

Die Laufzeiten der Prozesselemente B und C unterliegen ebenfalls einer Streuung, welche jedoch annähernd normalverteilt ist. Die Laufzeitverteilung der Prozesselemente ist in den Abbildungen A.16 und A.17 dargestellt. Der Mittelwert der Laufzeit von Prozesselement B beträgt 2min 51s bei einer Standardabweichung von 8s, die mittlere Laufzeit von Prozesselement C beträgt 9min 49s mit einer Standardabweichung von 32s.



Abbildung A.16: Laufzeitverteilung des Prozesselementes B

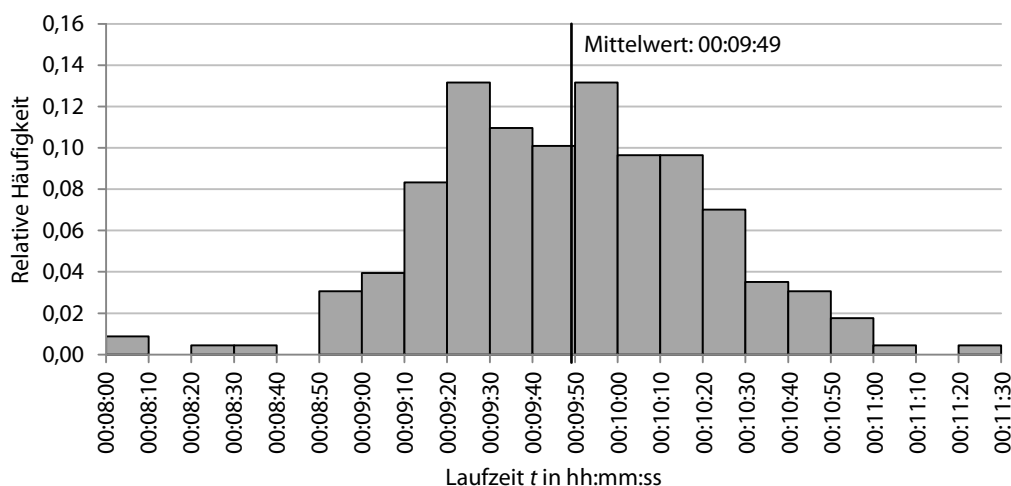


Abbildung A.17: Laufzeitverteilung des Prozesselementes C

Verantwortlich für die Streuung dieser Laufzeiten ist in erster Linie die unterschiedliche Modellgröße. Da bei der Vernetzung eines großvolumigen Modells mehr finite Elemente entstehen als bei einem kleinvolumigen Modell, müssen beim Solving mehr Gleichungen gelöst werden, wodurch sich eine längere Laufzeit ergibt.

A.7.2 Optimierungsergebnisse von Fallstudie 3

Bei der multidisziplinären Optimierung des Hebelmechanismus in Fallstudie 3 werden die folgenden Zustandsvariablen als Zielkriterien berücksichtigt:

- Minimierung der Masse m
- Minimierung der maximalen Verschiebung u_{max}
- Begrenzung der maximalen Spannung nach v. Mises auf $\sigma_{max} \leq 300$ MPa

Die im Prozesselement C berechneten Eigenfrequenzen (s. Abbildung A.14) werden bei der Optimierung nicht direkt als Zielkriterien verwendet. Die ersten beiden Eigenfrequenzen aller Individuen werden jedoch als Zustandsvariablen archiviert und dienen der Ergebnisanalyse und der Erhöhung der Systemverständnisses des Hebelmechanismus.

Bei der Formulierung der Zielfunktion werden die Attraktivitätskriterien Masse m und maximale Verschiebung u_{max} als gewichtete Zielkriterien summiert. Das KO-Kriterium der maximalen Spannung wird über eine externe Straffunktion implementiert. Die Zielfunktion wird durch die folgende Gleichung repräsentiert:

$$f(\mathbf{x}) = w_m \cdot \frac{m}{m_{max}} + w_u \cdot \frac{u}{u_{max}} + p_\sigma(\mathbf{x}) \text{ mit } w_m = 2 ; w_u = 1 ; m_{max} = 2 ; u_{max} = 10 \quad (\text{A.1})$$

Da die Zielkriterien m und u_{max} unterschiedliche Dimensionen besitzen, werden diese zunächst auf ihre jeweiligen maximalen Werte m_{max} und u_{max} normiert. Über die Gewichtungsfaktoren w_m und w_u wird die Gewichtung des jeweiligen Zielkriteriums gesteuert. Durch die in Gleichung (A.1) gewählten Werte, wird bei der Optimierung die Minimierung der Masse stärker verfolgt als die Minimierung der maximalen Verschiebung.

Die maximale Spannung nach v. Mises σ_{max} wird über die Straffunktion $p_\sigma(\mathbf{x})$ berücksichtigt, welche durch die folgende Gleichung abgebildet wird:

$$p_\sigma(\mathbf{x}) = \begin{cases} 0 & \sigma_{max} \leq 300 \text{ MPa} \\ 0,01 \cdot (\sigma_{max} - 300)^2 & \sigma_{max} > 300 \text{ MPa} \end{cases} \quad (\text{A.2})$$

Da der Strafterm p_σ erst einen Wert größer 0 annimmt, wenn der Grenzwert überschritten ist ($\sigma_{max} > 300$ MPa), wird diese Form der Straffunktion auch als externe Straffunktion bezeichnet (s. Abschnitt 2.8.1.3). Der Wert des Strafterms richtet sich danach, wie stark der Grenzwert überschritten wird. Die quadrierte Differenz bildet den Strafterm, welcher mit dem Skalierungsfaktor von 0,01 multipliziert wird. Dadurch werden geringe Überschreitungen des Grenzwertes nur wenig bestraft und die Individuen können im Verlauf der Optimierung ebenfalls zu guten Lösungen führen.

Das Ergebnis der Optimierung ist eine dreidimensionale Pareto-Front der bei der Optimierung evaluierten Individuen. Diese ist in Abbildung A.18 dargestellt. Zur Gewährleistung der Übersichtlichkeit wird eine zweidimensionale Darstellung der Attraktivitätskriterien m und u_{max} gewählt. Das KO-Kriterium σ_{max} wird über die Farbskala abgebildet.

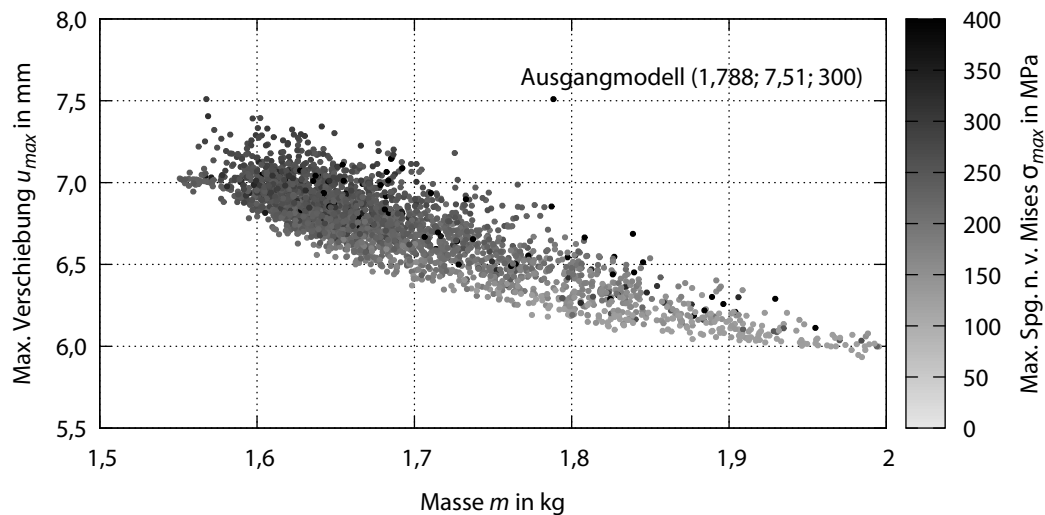


Abbildung A.18: Gegenüberstellung der Zielkriterien der Individuen des zweiten Optimierungslaufs

Anhand der Ergebnisse ist zu erkennen, dass die Individuen mit einer geringeren Masse zu einer größeren maximalen Verschiebung und einer größeren maximalen Spannung führen. Gegenüber dem Ausgangsmodell kann durch die Optimierung eine deutliche Reduzierung der Masse und der maximalen Verschiebung verzeichnet werden. Die Restriktion der maximalen Spannung wird dabei von den meisten Individuen eingehalten.

Die aus der Menge der Pareto-optimalen Lösungen gewählte Variante des Hebelmechanismus ist in Abbildung A.19 dargestellt. Dabei sind die optimierten Komponenten beschriftet. Diese Variante weist die folgenden Werte der Zustandsvariablen auf: $m = 1,557$ kg, $u_{max} = 6,98$ mm, $\sigma_{max} = 231$ MPa.

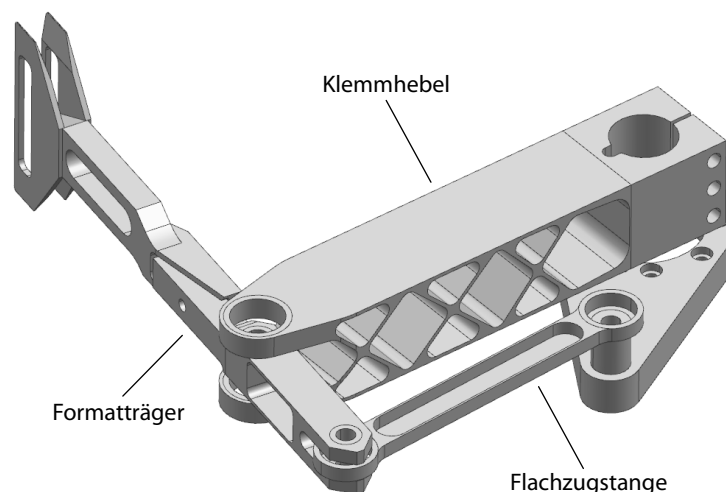


Abbildung A.19: Aus der Menge Pareto-optimaler Lösungen gewählte Variante des Hebelmechanismus

Bei der Optimierung wird ein GA mit einer Populationsgröße von $n_I = 80$ und einer Steady-State-Reproduktion verwendet (s. Abschnitt 2.6.2.1). Dabei werden pro Generation 50 Individuen erzeugt, welche die Individuen der Population ersetzen, sofern diese schlechter bewertet wurden. Dadurch wird bei der Optimierung sichergestellt, dass die Individuen der Population mindestens nicht schlechter, in der Regel jedoch stets besser werden, was bei der hier vorliegenden Minimierung zu einer kontinuierlichen Reduzierung des Zielfunktionswertes $f(\mathbf{x})$ führt. Dies ist gut zu erkennen, wenn die Individuen einer Generation nach ihrem Rang r_I angeordnet werden, wie in Abbildung A.20 dargestellt ist.

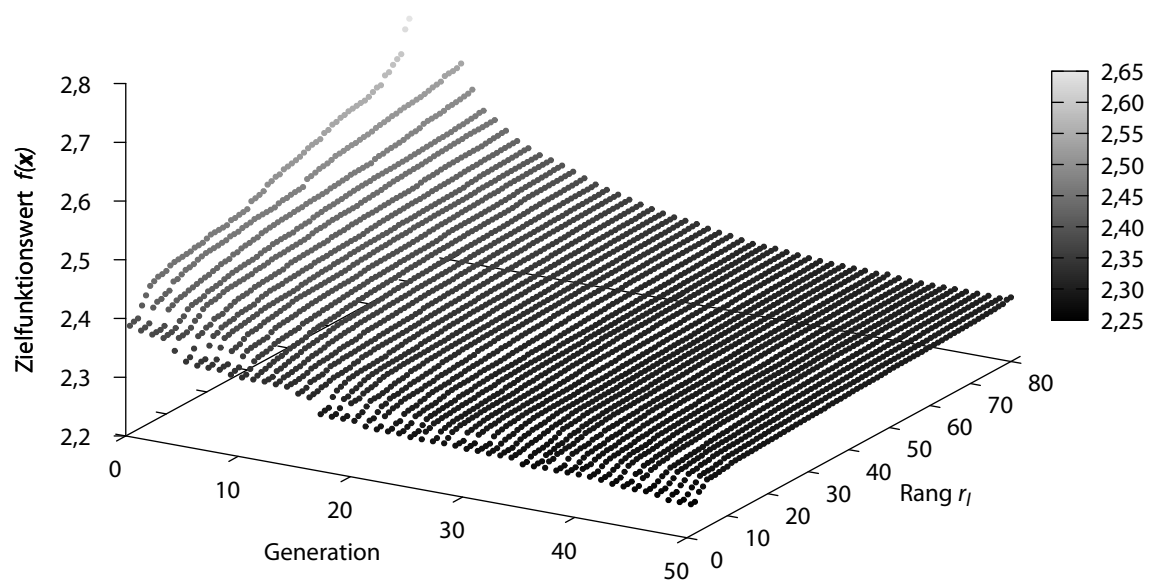


Abbildung A.20: Verlauf des Zielfunktionswertes und Rang der 50 besten Individuen einer Generation des zweiten Optimierungslaufs

Da die Zielkriterien bei einer multikriteriellen Optimierung in der Regel gegenläufig sind bzw. sich auch widersprechen können, werden zur Auswahl einer geeigneten Variante aussagekräftige und schnell erfassbare Darstellungen der relevanten Informationen benötigt. Die in Abbildung A.18 dargestellte Gegenüberstellung der Zielkriterien dient der Auswahl der Varianten anhand der Zielkriterien, gibt jedoch keine Information über die Designvariablen der optimalen Varianten. Dazu muss zunächst eine Variante ausgewählt, die Designvariablenwerte identifiziert und diese in den Designvariablenraum zurückgeführt werden (s. Abschnitt 2.4). Dies ist sehr aufwendig und keine intuitive Methode, die Zusammenhänge zwischen Designvariablen und Zustandsvariablen zu visualisieren.

Zum Aufbau bzw. zur Erhöhung des Systemverständnisses am Anschluss an die Optimierung ist es sinnvoll die Zusammenhänge zwischen Design- und Zustandsvariablen leicht verständlich und intuitiv zu visualisieren. Abi Akle et al. [AMY15] favorisieren hierzu die Darstellung mittels paralleler Koordinaten. In diesen Diagrammen wird jede Variable durch eine vertikale Achse repräsentiert und die Werte der Varianten durch verbundene Punkte dargestellt. Vor allem bei vielen Designvariablen (hier: $n = 33$) ist diese Darstellungsform unübersichtlich, da die Diagramme zu breit werden. Für die Darstellung der relevanten Zusammenhänge wird in dieser Fallstudie die Abbildung über Polardiagramme gewählt. Diese werden auch als Radar Charts oder Netzdiagramme bezeichnet.

Im Gegensatz zu den Polardiagrammen in [AMY15], werden hier in den Polardiagrammen ausschließlich die Werte der Designvariablen dargestellt und für jede Zustandsvariable ein separates Polardiagramm erstellt. Weiterhin werden nicht alle evaluierten Varianten abgebildet, sondern die Darstellung auf die jeweils besten Varianten beschränkt. Die Anzahl der Varianten kann hierbei vom Anwender festgelegt werden. Die Darstellung des Wertes der Zustandsvariable erfolgt über eine Farbskala. Die Polardiagramme können im Anschluss an eine Optimierung direkt aus DAG2OPT erzeugt werden. Dazu muss die Zustandsvariable, die Anzahl der darzustellenden Varianten und die Sortierreihenfolge ausgewählt werden.

Die Abbildungen A.21 – A.24 zeigen die Polardiagramme der jeweils besten 100 Varianten des Hebelmechanismus bezogen auf den Zielfunktionswert sowie die Zustandsvariablen Masse, max. Verschiebung und max. Spannung nach v. Mises. Jede Achse des Polardiagramms repräsentiert eine Designvariable. Die Bezeichnungen der Designvariablen stehen außen an den Achsen. Zusätzlich ist der Wertebereich in Klammern erfasst. Der Wertebereich der jeweiligen Designvariable ist auf den dargestellten Bereich der Achse normiert. Die untere Grenze des Wertebereiches wird auf der Hälfte der Achse angetragen, die obere Grenze außen.

Die Bezeichnung der Designvariablen erfolgt nach der Kurzform der Komponenten des Hebelmechanismus und der Art der Designvariablen. Dabei werden folgende Kürzel verwendet:

- ft: Formatträger
- kh: Klemmhebel
- fzs: Flachzugstange
- s: Wandstärke oder Abstand
- t: Tiefe einer Tasche
- n: Normierter Wert
- x: Breite der kreuzförmigen Rippenstruktur
- y: Abstand zwischen der kreuzförmigen Rippenstrukturen

Bei der Analyse des in Abbildung A.21 dargestellten Zusammenhangs zwischen den Designvariablen und den 100 besten Varianten bezogen auf den Zielfunktionswert ist zu erkennen, dass vor allem kleine Wandstärken (kh_s1 – kh_s12) und große Taschentiefen des Klemmhebels (kh_t1 – kh_t6) zu niedrigen Zielfunktionswerten führen.

Die Ähnlichkeit der Graphen in den Polardiagrammen von Zielfunktionswert und Masse (Abbildungen A.21 und A.22) zeigt den großen Einfluss der Masse in der Zielfunktion aufgrund des Gewichtungsfaktors von $w_m = 2$. Die Zusammenhänge zwischen den Designvariablen und der max. Verschiebung sowie der max. Spannung sind deutlich komplexer, wie in den Abbildungen A.23 und A.24 zu sehen ist.

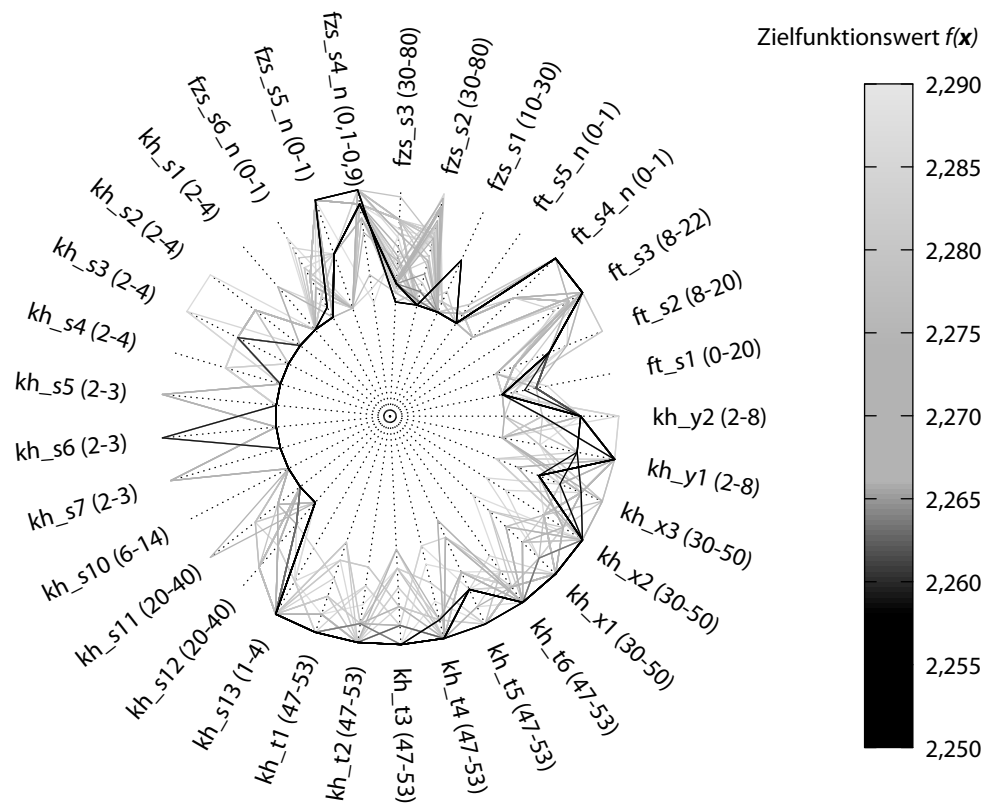


Abbildung A.21: Polardiagramm der Designvariablen der 100 besten Varianten bezogen auf den Zielfunktionswert

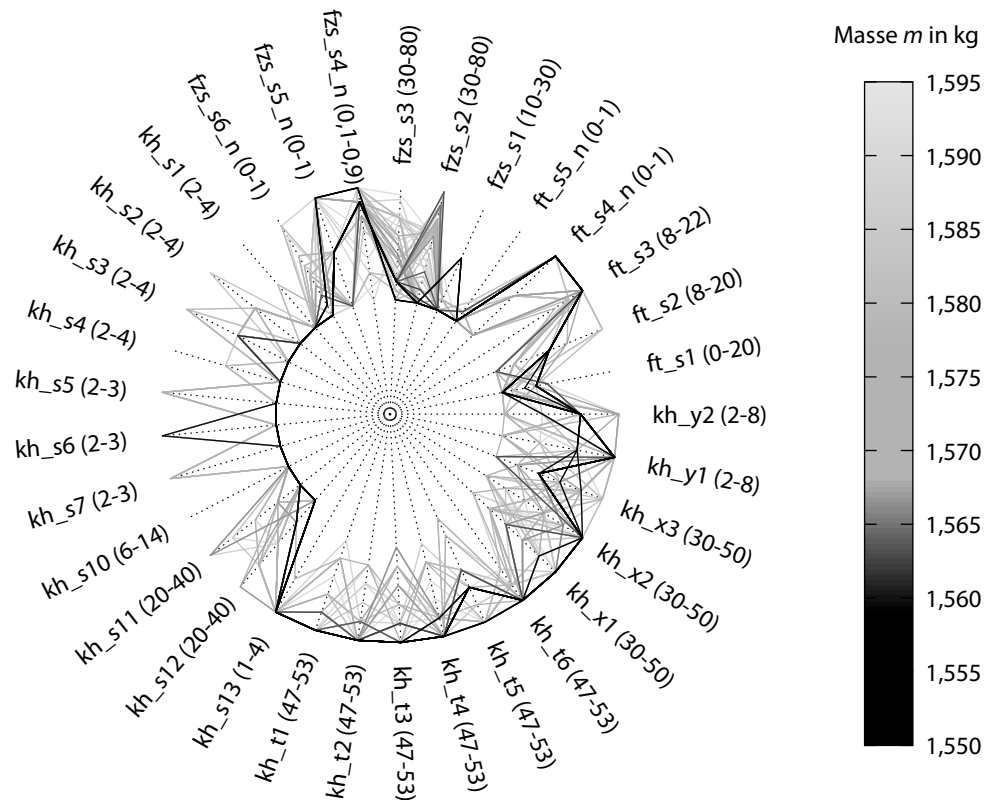


Abbildung A.22: Polardiagramm der Designvariablen der 100 besten Varianten bezogen auf die Zustandsvariable Masse

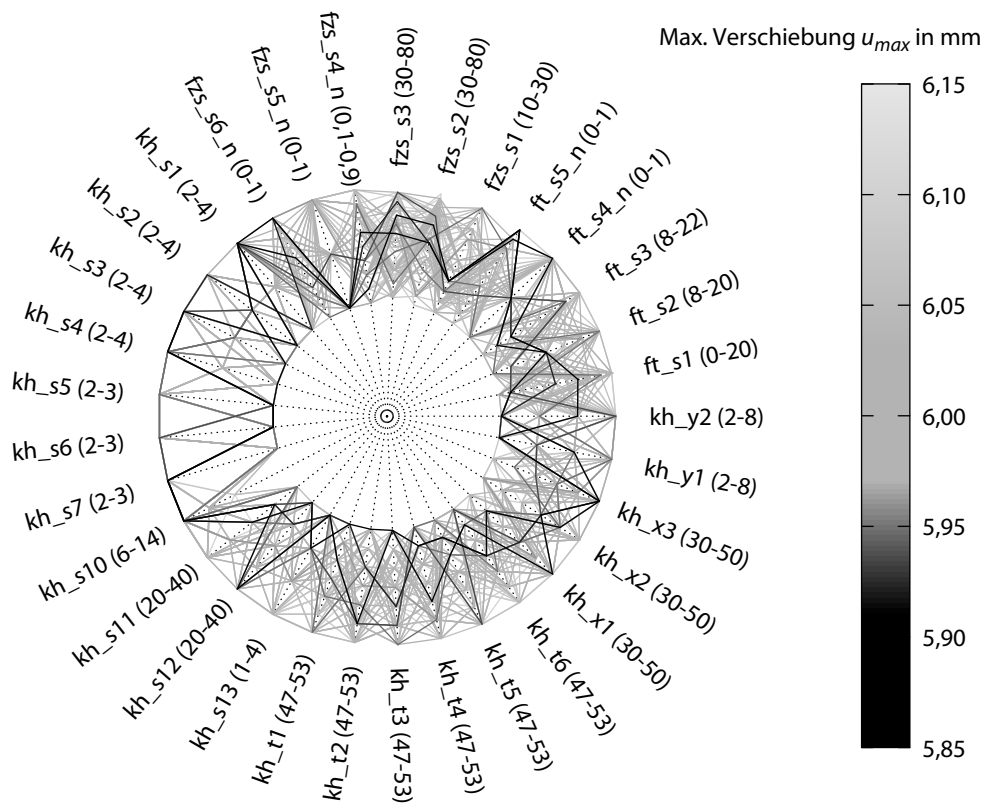


Abbildung A.23: Polardiagramm der Designvariablen der 100 besten Varianten bezogen auf die Zustandsvariable max. Verschiebung

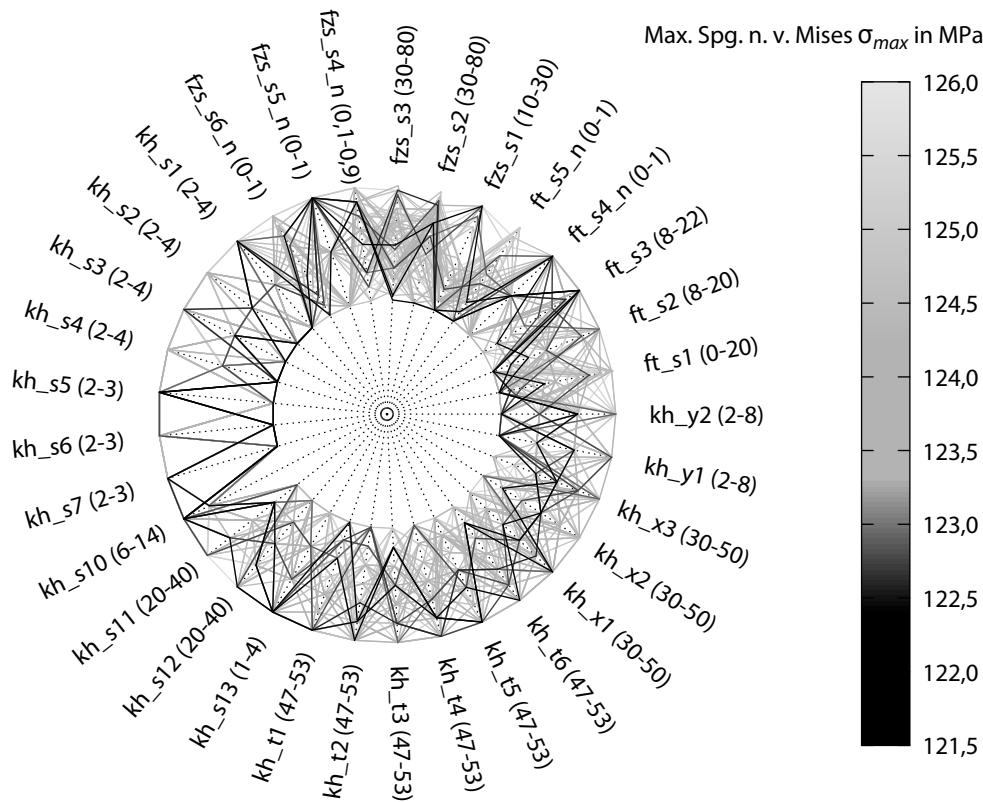


Abbildung A.24: Polardiagramm der Designvariablen der 100 besten Varianten bezogen auf die Zustandsvariable max. Spannung

A.8 Verwendeter Quellcode

A.8.1 Konfigurationsdatei des verwaltenden Rechners des Clusters

Der folgende Quellcode zeigt die zur prototypischen Umsetzung verwendete Konfigurationsdatei (*condor_config*) des verwaltenden Rechners (*Central Manager*) des HTCCondor-Clusters nach [Dzi15]. Auf die Standardeinstellungen von HTCCondor in der Präambel wird aufgrund der Übersichtlichkeit verzichtet.

```
1 # HTCCondor Mindestangaben
2 CONDOR_HOST = $(FULL_HOSTNAME)
3 COLLECTOR_NAME = cafe_pool
4 UID_DOMAIN = mb.uni-magdeburg.de
5 CONDOR_ADMIN =
6 SMTP_SERVER =
7 ALLOW_READ = *
8 ALLOW_WRITE = $(CONDOR_HOST), $(IP_ADDRESS), *.mb.uni-magdeburg.de
9
10 ALLOW_ADMINISTRATOR = $(IP_ADDRESS)
11 JAVA = C:\PROGRA~2\Java\JRE18~2.0_6\bin\java.exe
12 START = FALSE
13 WANT_VACATE = FALSE
14 WANT_SUSPEND = TRUE
15 DAEMON_LIST = MASTER SCHEDD COLLECTOR NEGOTIATOR KBDD ROOSTER
16
17 # Update-Intervalle
18 SCHEDD_INTERVAL = 30
19 NEGOTIATOR_UPDATE_INTERVAL = 30
20 MASTER_UPDATE_INTERVAL = 15
21 COLLECTOR_UPDATE_INTERVAL = 30
22 ROOSTER_INTERVAL = 120
23 UPDATE_INTERVAL = 30
24
25 # Concurrency Limits (Lizenzen)
26 NX10_LIMIT = 100
27
28 # Logs fuer den Energiesparplan (Hibernate)
29 OFFLINE_LOG = $(SPOOL)\OfflineLog
30 VALID_SPOOL_FILES = $(VALID_SPOOL_FILES), $(OFFLINE_LOG)
31 OFFLINE_EXPIRE_ADS_AFTER = 28800
32
33 # Automatisches Wake On LAN (WOL):
34 ROOSTER_UNHIBERNATE = Offline && Unhibernate
35 #ROOSTER_MAX_UNHIBERNATE = 10
36 ROOSTER_WAKEUP_CMD = "$(BIN)\condor_power.exe -d -i"
37
38 # Logs fuer den View Server (Statistiken)
39 POOL_HISTORY_DIR = C:\condor\ViewHist
40 KEEP_POOL_HISTORY = True
41 POOL_HISTORY_MAX_STORAGE = 100000000
42
43 # Globale RANK-Reihenfolge festlegen:
44 NEGOTIATOR_PRE_JOB_RANK = (MY.RANK)
```

A.8.2 Konfigurationsdatei des ausführenden Rechners des Clusters

Der folgende Quellcode zeigt die zur prototypischen Umsetzung verwendete Konfigurationsdatei (*condor_config*) des ausführenden Rechners (*Execute*) des HTCondor-Clusters nach [Dzi15]. Auf die Standardeinstellungen von HTCondor in der Präambel wird aufgrund der Übersichtlichkeit verzichtet.

```
1 # HTCondor Mindestangaben
2 CONDOR_HOST = PSG56
3 UID_DOMAIN = mb.uni-magdeburg.de
4 CONDOR_ADMIN =
5 SMTP_SERVER =
6 ALLOW_READ = *
7 ALLOW_WRITE = $(CONDOR_HOST), $(IP_ADDRESS), *.mb.uni-magdeburg.de
8 ALLOW_ADMINISTRATOR = $(IP_ADDRESS)
9 JAVA = C:\PROGRA~2\Java\JRE18~1.0_6\bin\java.exe
10 DAEMON_LIST = MASTER SCHEDD STARTD KBDD
11
12 # Update-Intervalle
13 MASTER_UPDATE_INTERVAL = 15
14 UPDATE_INTERVAL = 15
15 HIBERNATE_CHECK_INTERVAL = 300
16
17 # Vorhandene Software:
18 Calculix = False
19 NX10      = True
20 Python34 = True
21
22 # Gruppierung von Ressourcen
23 MachineOwner = "Dell-T1600"
24 STARTD_ATTRS = $(STARTD_ATTRS), MachineOwner, Calculix, NX10, Python34, ActTime
25 RANK = TARGET.Group =?= MY.MachineOwner
26
27 # Zeiten
28 # Ruhezeit der Maus und Tastatur, danach START Job
29 StartIdleTime = 5 * $(MINUTE)
30
31 # Ruhezeit der Maus und Tastatur, danach CONTINUE Job
32 ContinueIdleTime = 5 * $(MINUTE)
33
34 # Zeit, die Job rechnen darf, bevor er in SUSPEND geht
35 BeforeSuspendTime = 5 * $(MINUTE)
36
37 # Zeit, die Job max. im SUSPEND-Modus sein darf
38 MaxSuspendTime = 10 * $(MINUTE)
39
40 # Ruhezeit des Rechners, bevor er in den HIBERNATE-Modus geht
41 TimeToHibernate = 60 * $(MINUTE)
42
43 # Starten von Jobs
44 START = ( $(CPUIdle) && (KeyboardIdle > $(StartIdleTime)) )
45
46 # Allg. Einstellungen
47 WANT_SUSPEND = True
48 WANT_VACATE = False
```

```
49 # Pausieren von Jobs
50 SUSPEND = ( $(KeyboardBusy) && ($(StateTimer) > $(BeforeSuspendTime)) )
51
52 # Fortsetzen von Jobs
53 CONTINUE = ( $(CPUIIdle) && ($(ActivityTimer) > 10) \
54             && (KeyboardIdle > $(ContinueIdleTime)) )
55
56 # Entfernen von Jobs
57 PREEMPT = ( ((Activity == "Suspended") && \
58             ($(ActivityTimer) > $(MaxSuspendTime))) \
59             || (SUSPEND && (WANT_SUSPEND == False)) )
60
61 # Energiesparplan
62 ShouldHibernate = ( (KeyboardIdle > $(StartIdleTime)) \
63                    && $(CPUIIdle) && (State == "Unclaimed" && Activity == "Idle") \
64                    && ($(StateTimer) > $(TimeToHibernate)) )
65 HibernateState = "S4"
66 HIBERNATE = ifThenElse($(ShouldHibernate), $(HibernateState), "NONE")
67
68 # Ein Slot pro Rechner
69 SLOT_TYPE_1 = cpus=100%
70 NUM_SLOTS_TYPE_1 = 1
```

A.8.3 Jobdefinitionsdatei

Der folgende Quellcode zeigt beispielhaft eine vollständige Jobdefinitionsdatei für die Verwendung von HTCondor. Optionale Informationen sind auskommentiert.

```
1 #####
2 #
3 # HTCondor submit description file, Generated by DAG2OPT
4 #
5 #####
6
7 universe = vanilla
8
9 executable          = evaluation_1_update_preprocessing.bat
10 transfer_input_files = Scripts, Kurbelarm_links_sim1.sim, Kurbelarm_links_fem1.fem,
11                      Kurbelarm_links.prt, parameter.txt
12 #transfer_output_files =
13
14 log      = evaluation_1_update_preprocessing.log
15 output  = evaluation_1_update_preprocessing.out
16 error   = evaluation_1_update_preprocessing.err
17
18 should_transfer_files = yes
19 when_to_transfer_output = on_exit
20
21 #initialdir      = C:\path
22 #load_profile    = true
23
24 #on_exit_remove = (ExitBySignal == False) && (ExitCode == 0)
25
26 # Preference on Group Office
27 +Group = "Dell-T1600"
28 #+Group = "Dell-T5810"
29
30 # Concurrency Limits
31 concurrency_limits = NX10, Python34
32
33 # Requirements
34 #requirements = Arch=="X86_64" && OpSys=="WINDOWS"
35 requirements = (NX10 == True) && (Python34 == True)
36
37 # Rank
38 rank = memory
39
40 # Requests
41 #request_memory = 2048
42 #request_cpus   = 2
43 #request_disk   = 50000
44
45 # Priority
46 priority = 1
47
48 queue
```

A.8.4 DAGMan-Eingabedatei

Der Quellcode stellt beispielhaft eine DAGMan-Eingabedatei für die Verwendung von HTCCondor dar. Optionale Informationen sind auskommentiert.

```
1 #####
2 #
3 # HTCCondor submit DAGMan file, Generated by DAG2OPT
4 #
5 #####
6
7 JOB A evaluation_1_update_preprocessing.job
8 JOB B evaluation_2_solving_eigenwerte.job
9 JOB C evaluation_2_solving_linear_static.job
10
11 PARENT A CHILD B C
12
13 PRIORITY A 1
14 PRIORITY B 3
15 PRIORITY C 2
16
17 #SCRIPT POST D C:\script.bat $DAG_STATUS
18 #RETRY D 3
```

A.8.5 Post-Skript zur Fehlerbehandlung von Lizenzfehlern

Dieser Quellcode zeigt ein mögliches Post-Skript zur Fehlerbehandlung von Lizenzfehlern mit Hilfe der DAGMan-Wiederholungsanweisung nach [Dzi15].

```
1 @echo off
2 set suc=0
3 set err=1
4 find /i "The license server could be down" *.log
5 if %errorlevel%==1 (
6 exit /b %suc%
7 ) else (
8 sleep 300
9 exit /b %err%
10 )
```