Generative Layout- und Packingverfahren im digitalen Produktentwurf

Von der Fakultät
Luft- und Raumfahrttechnik und Geodäsie der Universität Stuttgart
zur Erlangung der Würde einer Doktor-Ingenieurin (Dr.-Ing.)
genehmigte Abhandlung

vorgelegt von
Claudia Schopper geb. Tonhäuser
geboren in
Schorndorf

Hauptberichter:	PD Dr.-Ing. Stephan Rudolph
Erster Mitberichter:	Assoc.Prof. Dipl.-Ing. Dr.techn. Mario Hirz
Zweiter Mitberichter:	Prof. Dr.-Ing. Andreas Strohmayer

Tag der mündlichen Prüfung: 31.10.2023

Institut für Flugzeugbau
Universität Stuttgart
2023

Danksagung

An dieser Stelle möchte ich meinen besonderen Dank nachstehenden Personen entgegen bringen, ohne deren Mithilfe die Anfertigung dieser Dissertation nicht zustande gekommen wäre:

Im Besonderen möchte ich mich bei meinem Doktorvater, Herrn PD Dr.-Ing. Stephan Rudolph, für das mir entgegengebrachte Vertrauen und die Betreuung der Arbeit bedanken. Mit seinem herausragenden Gespür für Forschungsdesign und Methodik führte er mich durch die komplexen Phasen meiner Arbeit, von der Themenfindung über die Umsetzung bis hin zur Qualitätskontrolle. Die zahlreichen Gespräche auf intellektueller und persönlicher Ebene werden mir immer als bereichernder und konstruktiver Austausch in Erinnerung bleiben. Ich habe unsere Dialoge stets als Ermutigung und Motivation empfunden.

Weiterhin möchte ich Herrn Assoc. Prof. Dipl.-Ing. Dr. techn. Mario Hirz für die hilfsbereite Übernahme des Mitberichts danken. Auch Herrn Prof. Dr.-Ing. Andreas Strohmayer möchte ich an dieser Stelle meinen Dank für die Übernahme des Mitberichts entgegenbringen.

Ein besonderer Dank geht an meine großartigen Kollegen der Pi-Gruppe. Vielen Dank für alle Gespräche, Diskussionen und Ratschläge. Ich habe mich stets geehrt gefühlt mit einem solch herzlichen, kreativen und exzellenten Team zusammenarbeiten zu dürfen. Meine Arbeit hat durch euch nicht unter Kollegen stattgefunden, sondern viel mehr unter Freunden.

Gleichermaßen möchte ich meinen Kollegen und Kolleginnen am Institut für Flugzeugbau an der Universität Stuttgart danken, welche mir jederzeit hilfsbereit zur Seite standen. An die gemeinsame Zeit am Institut erinnere ich mich immer mit Freude zurück. Ferner danke ich allen Kollegen und Kolleginnen des Forschungsprojekts „Zentrum für angewandte Forschung an Hochschulen angewandter Wissenschaft (ZAFH): Digitaler Produktlebenszyklus“. Unsere Zusammenarbeit im Projekt habe ich immer als wissenschaftlich bereichernd, kollegial und freundschaftlich wahrgenommen.

Von Herzen danken möchte ich meinen Eltern, Wilhelm und Hanna Tonhäuser, die mich während meines gesamten Studiums und der Arbeit unterstützt haben. Sie und meine Freunde haben während dieser intensiven Promotionszeit eine Säule der Stärke und des Frohsinns dargestellt. Euer unermüdliches Zuhören, eure ermutigenden Worte und die vielen Momente der Ablenkung, die wir miteinander teilten, waren unentbehrlich.

Tief verbunden und dankbar bin ich zuletzt meinem Ehemann, Dominik Schopper, dessen Liebe, Verständnis und unerschütterliche Präsenz eine Quelle des Trostes und der Inspiration war. Unsere langen gemeinsamen Abende mit fachlichen und persönlichen Diskussionen, sowohl in der Rolle meines Kollegen wie auch meines Partners, vor allem aber dein moralischer Beistand und dein liebevoller Ansporn sowie der grenzenlose Glaube an mich, haben mir Kraft und Mut zur Anfertigung und Vollendung meiner Dissertation gegeben.

Inhaltsverzeichnis

Kurzfassung

Der mit der Globalisierung einhergehende steigende Wettbewerbsdruck und eine nachhaltige Klima- und Energiepolitik stellen die nationale Industrie vor eine große Herausforderung. Um den Forderungen nach nachhaltigen, aber gleichzeitig kosteneffizienten Produkten von hoher Qualität gerecht zu werden, müssen immer komplexere Systeme entwickelt und hergestellt werden. Der steigenden Komplexität begegnen Unternehmen mithilfe von hierarchischen Lösungsprozessen und verteilten Aufgabenbereichen auf Entwicklungsteams mit komplementärem Hintergrund und Fachwissen. Diese haben jedoch häufig disparate Zielvorstellungen und unterschiedliche Entwicklungszeiten, was in der Praxis in inkompatiblen Modellen und Datenformaten, inkohärent weiterentwickelten Modellständen und inkonsistenter Datenhaltung mündet. Diese Missstände treten innerhalb des Produktentwurfes spätestens bei der Integrationsphase zutage und machen die Konfiguration inklusive der inhärenten Layout- und Packingaufgaben zu einer unübersichtlichen und daher manuell nur schwer zu bewältigenden Herausforderung. In vollem Umfang ist diese Problematik ausschließlich durch eine vollständige Formalisierung und Digitalisierung des Produktentwurfes zu bewältigen. Über die notwendige konsistente Datenhaltung hinaus ermöglicht eine solche digitale Repräsentation außerdem eine Automation des Entwurfsprozesses, was mit einer Verkürzung von Entwicklungszeiten einhergeht und damit zur Steigerung der Produktivität und Effizienz führt. Vor diesem Hintergrund besteht ein dringender Bedarf an effektiven Vorgehensweisen für die Formalisierung von Entwurfswissen sowie an digitalen Methoden zur Repräsentation, Unterstützung und Automatisierung der Entwurfsaufgaben innerhalb des Produktentwicklungsprozesses.

Die vorliegende Arbeit ist dem Vorhaben gewidmet, eben diese Methoden und adäquate digitale Hilfsmittel für die Integrationsphase des Produktentwurfes bereitzustellen. Im Mittelpunkt steht dabei die Entwicklung eines Frameworks auf Basis eines fundamentalen, ganzheitlichen und systemorientierten Ansatzes. Das Framework soll den Lösungsprozess bei Layout- und Packingaufgaben mithilfe von wissenschaftlich fundierten, systematischen und flexiblen Entwurfsmethoden unterstützen. Dafür soll es die Möglichkeit bieten, durch eine effiziente computergestützte Generierung und Bewertung von Entwurfslösungen neue Konfigurationen automatisiert zu erforschen und somit die Entwicklung von fortschrittlicheren und komplexeren Systemen ermöglichen. Die graphenbasierten Entwurfssprachen dienen hierbei aufgrund ihrer wissenschaftlichen und einheitlichen Entwurfstheorie als Basis für die Wissensrepräsentation und als Plattform zur Implementierung. Im Laufe der vorliegenden Arbeit kann gezeigt werden, dass das entwickelte und implementierte Framework auf Problemstellungen aus der Konfigurations- und Integrationsphase für Anwendungsfälle in unterschiedlichsten Domänen eingesetzt werden kann. Darüber hinaus kann demonstriert werden, dass Anwenderinnen und Anwender bei Layout- und Packingaufgaben entlastet werden, da die im Framework bereitgestellten Instrumente die Lösungsfindung durch Teilautomatisierung und Optimierung unterstützten. Die vorliegende Arbeit legt damit den Grundstein zur Entwicklung eines umfassenden automatisierten Entwurfsverfahrens, welches unabhängig von einer Domäne Integrationsaufgaben übernehmen kann und geht einen weiteren Schritt in Richtung vollständiger Automatisierung des Produktentwicklungsprozesses.

Abstract

The increasing competitive pressure associated with globalization and sustainable climate and energy policies pose a major challenge to national industry. In order to meet the demands for sustainable, but at the same time cost-efficient products of high quality, increasingly complex systems must be developed and manufactured. Companies counter the increasing complexity with the help of hierarchical solution processes and distributed task areas to development teams with complementary backgrounds and expertise. However, these teams often have disparate objectives and different development times, which in practice results in incompatible models and data formats, incoherently developed model states and inconsistent data management. These grievances become apparent within the product design at the latest during the integration phase and make the configuration, including the inherent layouting and packing tasks, a confusing challenge that is therefore difficult to master manually. This problem can only be fully overcome by completely formalizing and digitizing the product design. Beyond the necessary consistent data management, such a digital representation also enables automation of the design process, which goes hand in hand with a reduction of development times and thus leads to an increase in productivity and efficiency. Against this background, there is an urgent need for effective procedures for the formalization of design knowledge as well as for digital methods for the representation, support and automation of design tasks within the product development process.

The present work is dedicated to the project of providing these methods and adequate digital tools for the integration phase of product design. The focus is on the development of a framework based on a fundamental, holistic and system-oriented approach. The framework shall support the solution process for layout and packaging tasks with the help of scientifically based, systematic and flexible design methods. For this purpose, it shall provide the possibility to explore new configurations in an automated way through an efficient computer-aided generation and evaluation of design solutions, thus enabling the development of more advanced and complex systems. Here, graph-based design languages serve as a basis for knowledge representation and a platform for implementation due to their scientific and unified design theory. In the course of the present work, it can be shown that the developed and implemented framework can be applied to problems from the configuration and integration phases for use cases in a wide variety of domains. Furthermore, it can be demonstrated that users are relieved of layout and packaging tasks, since the tools provided in the framework supported solution finding through partial automation and optimization. The present work thus lays the foundation for the development of a comprehensive automated design procedure that can perform integration tasks independently of a domain and takes a further step towards the complete automation of the product development process.

KAPITEL 1

Einleitung

Der globale Wettbewerb bei gleichzeitiger Verbesserung internationaler Produkte, steigende Energiepreise und der dringende Bedarf an umweltorientiertem Handeln sind nur einige der zu bewältigenden Aufgaben der heutigen Industrie. Gegenwärtig sind Unternehmen mit einer Situation konfrontiert, in der eine immer lauter werdende Forderung nach technischen Systemen mit steigender Energieeffizienz und günstigerem Emissionsverhalten gestellt wird, welche umweltschonend sind und gleichzeitig die Stärkung der wirtschaftlichen Wettbewerbsfähigkeit sicherstellen. Der heutige Systementwurf steht infolgedessen ebenso vor der Herausforderung, diese Ansprüche umzusetzen. Da der Entwurf eines technischen Systems und dessen Subsystemen und Komponenten eine hohe Multidimensionalität und gegenseitige Interaktion der beteiligten Elemente aufweist, handelt es sich hierbei um eine hochkomplexe Aufgabe, in der die inhärente Hierarchie im System und die Vielschichtigkeit aller beteiligten Disziplinen konsistent abgebildet und stets im Auge behalten werden muss. In der Praxis wird die hohe Komplexität mithilfe von hierarchischen Lösungsprozessen und verteilten Aufgabenbereichen bewältigt. Beispielsweise wird der Subsystem- und Komponentenentwurf, welcher spezialisiertes Wissen aus einer oder mehreren Disziplinen erfordert, in der Regel unabhängigen Entwicklungsteams mit komplementärem Hintergrund und Fachwissen zugewiesen. Diese bedienen sich unterschiedlichen digitalen Hilfswerkzeugen und weisen unterschiedliche, teilweise sogar konträre Teilziele auf. Die ungleichen Zielsetzungen spiegeln sich in differierenden Lösungsmengen wider, welche mithilfe von problemangepassten Lösungsstrategien und Lösungsalgorithmen gefunden werden. Eine weitere Erschwernis bei der Aufspaltung des Gesamtentwurfes in Teilaufgaben sind die unterschiedlichen Entwicklungszeiten, die für die jeweiligen Systeme und Komponenten aufgebracht werden müssen. All diese Faktoren führen zu inkohärent weiterentwickelten Modellständen und inkonsistenter Datenhaltung innerhalb des Produktentwurfes. Trotz der im heutigen Systementwurf vorherrschenden entkoppelten Verteilung der Entwurfsaufgaben weisen die einzelnen Subsysteme und Komponenten jedoch zahlreiche Wechselwirkungen auf. Während jede Subsystemoptimierung im Hinblick auf ihre ganz eigenen Zielfunktionen durchgeführt wird, beeinflusst ihre Entwurfslösung die räumlichen Konfigurationen aller anderen Subsysteme und vice versa. Die Kopplungen werden in der gegenwärtigen Praxis häufig erst im nachfolgenden Integrations- bzw. *Layout-* oder *Packing*-Prozess berücksichtigt, welcher mithilfe von Entwurfsregeln, Erfahrungen aus vergangenen Entwürfen und der individuellen Intuition von Experten manuell gelöst werden muss. Dieses Vorgehen bringt zwar durchaus realisierbare Entwürfe hervor, diese sind jedoch möglicherweise nicht optimal hinsichtlich aller Systemanforderungen und die Komplexität der Systeme, die in Betracht gezogen werden können, ist begrenzt [Peddada et al., 2021b]. Aus diesem Grund müssen Layout- und Packingprobleme im verteilten und dezentralisierten Produktentwurf mithilfe adäquater digitaler

Methoden zur Entwurfsautomation unter Berücksichtigung der inhärenten Kopplungen der Einzelsysteme gelöst werden. Die explizite Behandlung der komplexen Entwurfskopplungen mittels ganzheitlicher Entwurfsoptimierungsmethoden ist der Schlüssel, um den gegenwärtigen und zukünftigen Herausforderungen bei der Integration von komplexen Systemen gerecht zu werden. Einer der wichtigsten Vorteile eines solchen Vorhabens ist die Verringerung der Entwurfszeit und eine damit einhergehende Verringerung des Entwurfsaufwands bei gleichzeitiger Entwurfsoptimierung und damit einhergehender Schonung der Ressourcen, welche für Herstellung und Betrieb notwendig sind.

Der Erhaltung der wirtschaftlichen Stärke auf eine umweltorientierte Weise und die daraus resultierende Notwendigkeit für die Digitalisierung und Entwurfsautomation reicht ferner über den technischen Systementwurf hinaus. Auch die Architektur als enger Partner des Bauingenieurwesens wird immer mehr von den digitalen Fortschritten bestimmt und es wird inzwischen anerkannt, dass digitale Entwurfsmethoden auch hier bald eine Standardmethode sein werden, um die menschliche Fähigkeit zu erweitern, „das Unsichtbare zu sehen und das Ungedachte zu gestalten“ [Zawidzki und Szklarski, 2020]. Die Verwendung von Algorithmen gibt also nicht nur Ingenieuren und Ingenieurinnen, sondern auch Architekten und Architektinnen einen völlig neuen Werkzeugkasten mit an die Hand, mit dem sie ihre Ideen umsetzen und verbessern können [Carta, 2020]. Bedauerlicherweise sind das Ingenieurwesen und die Architektur seit jeher zwei unabhängige Forschungsdisziplinen. Historisch gesehen sind diese Gemeinschaften sogar fast vollständig disjunkt: Sie haben unterschiedliche Forschungseinrichtungen, unterschiedliche Zeitschriften und unterschiedliche Anwendungsfälle. Dennoch basieren ihre Aufgabenstellungen in vielen Bereichen auf denselben Grundproblemen, sodass sich beide Domänen gegenseitig bereichern würden. Heute arbeiten immer mehr Architekten und Architektinnen mit Computerwissenschaftlern und -wissenschaftlerinnen zusammen, um neue Wege für die automatisierte Generierung von effizienten Lösungen zu finden. Es wird Zeit, dass auch das Ingenieurwesen mit der Architektur zusammen kommt, um gemeinsame Verfahren zu entwickeln. Die Bedeutung einer engen Zusammenarbeit der beiden Fachbereiche wird auch in [Beghini et al., 2014] erörtert. Ansätze zur Entwicklung von Informationstechnologien für die Automatisierung von Layout- und Packingaufgaben sollten eine Brücke zwischen beiden Disziplinen schlagen und sich einem generischen und domänenunabhängigen Ansatz widmen, welcher von beiden Branchen gleichermaßen inspiriert wird.

Die Entwicklung und Umsetzung von intelligenten Informationstechnologien für die Synthese räumlicher Konfigurationen komplexer technischer Systeme mit materiellen Systemkomponenten unter Berücksichtigung ihrer räumlichen Form und zusätzlicher geometrischer und physikalisch-mechanischer Zwänge ist dementsprechend losgelöst von einer speziellen Disziplin, und ist Thema in vielen Wissenschaften. Domänenübergreifend findet sie bei Forschern gegenwärtig immer mehr Beachtung. Die Weiterentwicklung eines systemorientierten Ansatzes erfordert dabei Formalismen und Methoden, die noch nicht existieren. Tiefgreifende Fortschritte konnten hierbei nur selten erzielt werden, was zum Teil auf den Mangel an geeigneten konsistenten Entwurfsrepräsentationen zurückzuführen ist, die mit potenziellen Entwurfsautomatisierungsstrategien kompatibel sind, genauso wie sie an dem Fehlen einer einheitlichen Entwurfstheorie scheitern [Peddada et al., 2021b].

Die vorliegende Arbeit ist dieser Thematik gewidmet. Im Mittelpunkt der Arbeit steht die Entwicklung eines Frameworks auf Basis des fundamentalen ganzheitlichen und systemorientierten Ansatzes der graphenbasierten Entwurfssprachen, um die automatisierte Lösung von Layout- und Packingproblemen mithilfe von systematischen, flexiblen und wissenschaftlich fundierten Entwurfsmethoden zu unterstützen, welches für verschiedenste technische Branchen

die Möglichkeit bietet, durch eine effiziente Generierung und Bewertung von Entwurfslösungen neue Konfigurationen zu erforschen und dadurch eine fortschrittlichere Systementwicklung zu ermöglichen. Das entwickelte Framework soll praktizierenden Systementwicklern und -entwicklerinnen ermöglichen, über das hinauszugehen, was mit bestehenden Methoden bisher möglich war und die realisierbare Komplexität von Systemen steigern, sodass eine höhere Systemleistung und -funktionalität bei geringerem Ressourceneinsatz erzielt werden kann. Der Leser wird hierbei beginnend mit der Definition des Layout- und Packingproblems und dem Vorschlag für einen Lösungsansatz in *2 Problemspezifikation und Lösungsansatz* durch die aktuellen Methoden in *3 Stand der Technik im Bereich des automatisierten Packings* geführt und erfährt in *4 Ein Framework für Layout und Packing in graphenbasierten Entwurfssprachen* wie das vorgeschlagene Framework aufgebaut ist und implementiert wurde. Während in *5 Packingentwurfssprachen in Anwendung* Probleme vorgestellt werden, welche mittels des Frameworks gelöst werden können, lässt die Arbeit die gewonnenen Ergebnisse in *6 Zusammenfassung & Fazit* Revue passieren und schließt mit einem Ausblick in die Zukunft ab.

KAPITEL 2

Problemspezifikation und Lösungsansatz

Das Problem des räumlichen zweidimensionalen Layouts oder dreidimensionalen Packings von komplexen vernetzten Systemen kann als optimale räumliche Anordnung heterogener geometrischer Komponenten und deren Kompositionen von oft nicht trivialer Form in unregelmäßigen Gebieten oder Bauräumen unter Berücksichtigung ihres physikalischen Verhaltens, der lebenszyklusbezogenen Prozesse und der Systembetriebsbedingungen beschrieben werden [Peddada et al., 2021b]. Basierend auf [Peddada et al., 2021b] kann das räumliche Packing eines Gesamtsystems in vier grundlegende Teilprobleme unterteilt werden:

1. Die Anordnung von Systemkomponenten bzw. deren Layout oder Packing.
2. Die Ausprägung der Komponentenvernetzung in Form einer Bauteilverkabelung und -verrohrung oder in Form von Verkehrs- und Transportwegen.
3. Eine physikbasierte Dimensionierung der einzelnen Komponenten sowie die physikbasierte Konfigurations- bzw. Topologieoptimierung des Gesamtsystems.
4. Die digitale Repräsentation des ganzheitlichen Packingprozesses und -ergebnisses.

Die Anordnung von Systemkomponenten bzw. deren zweidimensionales Layout oder dreidimensionales Packing gehört zur Klasse der mathematischen Optimierungsprobleme, bei denen es um die optimale Platzierung und Ausrichtung von Objekten innerhalb eines gegebenen Gebiets oder Bauraums auf der Grundlage einer Reihe von Bewertungskriterien geht, oder um die Minimierung des Bauraumvolumens bzw. der Gebietsfläche, in oder auf dem die Objekte unter Berücksichtigung einiger Entwurfsziele und -beschränkungen untergebracht werden können. Typische Anwendungsfälle sind technische Systeme als Kombination aus funktional und geometrisch zusammenhängenden Komponenten aus dem Ingenieurwesen, die Fragestellungen können aber genauso auch auf Systeme aus dem Bauwesen übertragen werden. Die zu optimierende räumliche Lage und Ausrichtung von Komponenten beeinflussen eine Reihe physikalischer Größen, die für den Entwickler, den Konstrukteur, den Hersteller und den Endverbraucher des Produkts von Interesse sind [Peddada et al., 2021b]. Die wichtigsten Einschränkung beim Packing sind in nahezu allen Fällen, dass sich die Komponenten nicht überschneiden und dass sie gleichzeitig nicht aus dem Entwurfsraum herausragen. Weitere Nebenbedingungen können die räumlichen Beziehungen zwischen Komponenten sein, geometrische Verbotszonen, Abstände zueinander oder zu vorher definierten Punkten, die Sicherstellung der Zugänglichkeit für Wartungsarbeiten, die Sicherstellung der Versorgung und viele mehr.

Die Ausprägung der Komponentenvernetzung in Form einer Bauteilverkabelung und -verrohrung oder in Form von Verkehrs- und Transportwegen umfasst die materielle Ausprägung der

Schnittstellen zwischen den Systemelementen auf Systemebene. Sie beeinflusst die Systemtopologie nicht auf funktionaler Ebene, da sie keinen Einfluss auf die Schnittstellen selbst nimmt, allerdings kann sie die geometrische Topologie des Gesamtsystems beeinflussen, wenn die Verbindungselemente als geometrische Elemente in das Gesamtsystem eingehen. Bewertungskriterien sind hierbei häufig die Gesamtlänge oder das Gesamtgewicht der Verbindungen oder physikalische Größen, welche Auskunft über die Effizienz der ausgeprägten Verbindungswege geben. Wird die Ausprägung der Komponentenvernetzung nicht erst im Nachgang zur Elementpositionierung und -orientierung ausgeführt, ist die Bestimmung der Platzierung auf natürliche Weise ebenso eine Teilaufgabe bei der Ausprägung der Vernetzung, da sie das Hauptkriterium für die Optimalität der Gesamtbewertung darstellt.

Im Sinne eines ganzheitlichen Ansatzes soll davon ausgegangen werden, dass die Maße der einzelnen Elemente zum Zeitpunkt eines Packingprozesses nicht zwingend eingefroren sind und zum Vorteil eines Gesamtpackingentwurfs manipuliert werden können. Die Dimensionierung der Systemkomponenten beschreibt dabei deren Parametrisierung und muss als Teilproblem in den Packingprozess eingehen. Weiterhin muss ebenso deren Konfiguration in Form einer Topologieoptimierung als Teilproblem berücksichtigt werden. Die Topologieoptimierung bezieht sich in diesem Kontext auf die Optimierung der Systemtopologie, welche alle Komponenten und deren materielle Verbindungen umfasst. Sie geht von der Systemarchitektur aus, welche sowohl die funktionalen wie auch die materiellen Schnittstellen der Komponenten untereinander beschreibt, und bildet die Basis für die anschließenden räumlichen Ausprägung der gewünschten Schnittstellen in Form von verbindenden Elementen.

Die wichtigste Aufgabe eines ganzheitlichen Packings ist abschließend, die genannten Teilprobleme, unter Berücksichtigung aller geometrischer und physikalischer Wechselwirkungen zwischen den Systemkomponenten, in einer konsistenten Problemdarstellung zu integrieren, wofür das letzte der genannten Teilprobleme, also eine digitale Repräsentation des ganzheitlichen Packingprozesses und -ergebnisses, benötigt wird.

2.1 Herausforderungen im Layout- und Packingprozess

Das ebene und räumliche Packing von komplexen vernetzten Systemen stellt Systemarchitekten und -architektinnen also vor eine Reihe wesentlicher Aufgaben und Herausforderungen. Werden die Layout- oder Packingprozesse aus verschiedenen technischen Systemen untersucht, können eine handvoll Forschungsschwerpunkte für die Entwicklung eines automatisierten Packingprozesses identifiziert werden, die fast allen Packingprozessen gemein sind.

Der erste Kernpunkt umfasst dabei die Definition des eigentlichen Packingziels und der Entwicklung und Formalisierung einer Bewertungsfunktion, welche über eine optimale Lösung entscheidet. Die Definition der meistens multikriteriellen Bewertungsfunktion findet vorwiegend über die Modellierung einer Zielfunktion als gewichtete Summe aus Designzielen und optional Strafen für die Verletzung von Randbedingungen statt, oder in Form eines Paretooptimums. Zu den Entwurfszielen kann die Quantifizierung einer Vielzahl von Kriterien gehören, z. B. die Menge der benötigten Kabel zwischen elektronischen Bauteilen, die Strömungsleistung der notwendigen Verrohrung, die Packungsdichte eines Systems aus Komponenten, der Massenschwerpunkt des Gesamtproduktes, das Gesamtvolumen bzw. die Gesamtfläche, die Ausleuchtung eines Gebäudes oder viele andere. Die Kunst bei der Modellierung einer Zielfunktion ist, die zu optimierenden Kriterien geeignet zu repräsentieren, sie also mathematisch so zu erfassen, dass einerseits das zu untersuchende technische Problem zweckdienlich abgebildet wird und dass andererseits eine möglichst eindeutige beste Lösung existiert, die sich zudem

noch in einer dem technischen Problem angepassten Rechenzeit lösen lässt [Groell, 2018]. Als weitere Herausforderung muss die modellierte Zielfunktionen im Sinne einer Bewertungsfunktion außerdem einen sowohl qualitativ als auch quantitativ sinnvollen Vergleich zwischen den möglichen Lösungsvarianten sicherstellen, sodass für das Problem möglichst objektive Optima gefunden werden können.

Eine verwandte Aufgabe, die es als zweite Herausforderung zu lösen gilt, liegt in der Abbildung und Modellierung der Anforderungen, Randbedingungen und Entwurfszwänge. Die häufigsten Nebenbedingungen, welche bei Packingprozessen Anwendung finden, sind geometrischer Natur, beispielsweise wenn räumliche Beziehungen zwischen Bauteilen berücksichtigt werden, oder vorgegebene Abstände eingehalten werden müssen. Aber nicht nur geometrische Aspekte sind zu berücksichtigen, auch physikalische Eigenschaften wie das Gewicht, Temperaturen und thermische Belastungen, hydraulische Aspekte, genauso wie aerodynamische und elektromagnetische Einflüsse spielen wesentliche Rollen bei der Integration von Gesamtsystemen. Dabei ist zu berücksichtigen, dass sich die Bedingungen nicht nur auf die Einzelkomponenten eines Systems beziehen, sondern viel mehr auf das ganzheitliche Gesamtsystem. Um solche Bedingungen abbilden zu können, wird ein Framework benötigt, welches ermöglicht, die verschiedenen Domänen gleichzeitig einzufangen und konsistent zu verarbeiten, sowohl für die Einzelsysteme wie auch für das Gesamtprodukt. Auch [Peddada et al., 2021b] stellt fest, dass die Erstellung von flexiblen Entwurfsrepräsentationen, welche Topologie-, Geometrie- und Physikaspekte des Gesamtproblems auf einheitliche Weise unterstützen, eine der größten Schwierigkeiten im automatisierten Packing darstellt.

Eine der universellsten Nebenbedingungen bei Packingprozessen ist die Kollisionsvermeidung, welche sicherstellt, dass Objekte niemals denselben Standort einnehmen oder überlappen. Dies wird als sog. Interferenzprüfung bezeichnet und findet sich in nahezu jedem Packingproblem. Es gibt drei grundlegende Wege, um eine Interferenzerkennung umzusetzen. Wird eine Oberflächenrepräsentation der Objektgeometrien verwendet, müssen Algorithmen entwickelt werden, welche ein Durchdringen zweier Oberflächen zuverlässig und effizient berechnen. Häufig stellt die Berechnung der tessellierten Objekte jedoch eine rechnerische Herausforderung dar, vor allem bei nicht konvexen Objekten mit Hohlräumen und Löchern. Sollen zur Interferenzreduktion zusätzliche Informationen verwendet werden wie die Eindringtiefe und die Bewegungsrichtung, werden die Routinen inakzeptabel rechenaufwändig und liefern im Extremfall überhaupt keine Lösung [Fadel und Wiecek, 2015]. Eine andere Möglichkeit ist eine vereinfachte, volumetrische Darstellung der Objektgeometrie. Dabei kann die Geometriepräsentation in eine Gruppe von einfachen geometrischen Untereinheiten unterteilt werden, für die jeweils eine Kollisionsüberprüfung stattfinden muss. Handelt es sich um eine regelmäßige Unterteilung (wie z.B. eine Voxelisierung), kann der Aufwand einer Interferenzüberprüfung deutlich reduziert werden. Die dritte Variante der Interferenzprüfung stellt die abstrakte Darstellung der Geometrien inklusive deren Positionen und Orientierungen über implizite mathematische Gleichungen und Ungleichungen dar. Mithilfe mathematischer Techniken können die aufgestellten (Un-) Gleichungssysteme gelöst und somit Kollisionen vermieden werden. Häufig können dieselben Formulierungen gleichzeitig für die Definition von weiteren Randbedingungen wie Kontakt zu einem Körper, eine gewünschte Verschneidung zwischen Körpern oder die Inklusion eines Körpers in einem anderen verwendet werden (z.B. in [Chernov et al., 2010]). Je komplexer die Geometriepräsentation hierbei ist, desto komplexer sind die Gleichungssysteme in Bezug auf Grad, Dimension (Anzahl der Freiheitsgrade) und Anzahl der Gleichungen.

Die Erkennung von Interferenzen bzw. Kollisionsdetektion steht also in direktem Zusammenhang mit der Wahl der Geometriepräsentation. Dies führt zum dritten Kernproblem, der

Wahl einer geeigneten geometrischen Darstellung. Im allgemeinen Fall können die Komponenten eines Packingproblems von beliebig komplexer geometrischer Gestalt sein. Ein hoher Komplexitätsgrad wird dann extrem schnell erreicht, wenn die Geometrien unregelmäßige Formen aufweisen, besonders bei Nichtkonvexität. Sollen die Maße des umhüllenden Gehäuses minimiert werden oder ist das Gehäuse klein gewählt, muss ein besonders enges Packing berechnet werden. Unter diesem Gesichtspunkt ist eine genaue Untersuchung des Suchraums erforderlich, was eine möglichst genaue Repräsentation der Geometrie notwendig macht. Andererseits ist es für eine effiziente Berechnung oft wünschenswert, einfache Geometriedarstellungen zu haben, deren Überlappungs- oder Kollisionstests schnell sind [Cagan et al., 2002], was weitgehend im Widerspruch zur möglichst genauen Geometrierepräsentation steht, da dies für eine möglichst einfache Approximation der Geometrie spricht.

Die vierte Kernfrage befasst sich mit einer geeigneten Repräsentation des Lösungsraumes und der Freiheitsgrade. Im einfachsten Fall werden in einem Packingprozess die translatorischen und die rotatorischen Freiheitsgrade von vorgegebenen Systemkomponenten bestimmt. Wird das Packing als Teil des Gesamtentwurfs interpretiert, welcher rückwirkend auf vorherige Auslegungsschritte Einfluss nehmen kann, kann die Veränderung der Bauteilgeometrie in Form einer Skalierung als weiterer Freiheitsgrad eingehen. Je nach verfügbarer Rechenleistung gibt es verschiedene Ansätze, um Freiheitsgrade zu repräsentieren, diese können kontinuierlich oder diskret sein und bei Bedarf einen Wertebereich zugewiesen bekommen. Dabei ist nicht unbedingt gesagt, dass die Diskretisierung des Lösungsraumes zu effizienteren Ergebnissen führt. Zwar ist die Varianz bei einem kontinuierlichen Wertebereich theoretisch höher als bei einer endlichen Anzahl von diskreten Werten, stetige Funktionen lassen jedoch eine breitere Bandbreite an möglichen Optimierungsverfahren zu. Um den möglichen Lösungsraum einzuschränken, werden dennoch häufig diskrete Variablen verwendet. Die gewünschte Folge ist eine Einsparung der Rechenzeit, diese geht jedoch auf Kosten der Optimalität, werden dadurch immerhin Teile des Entwurfsraumes von vorneherein ausgeschlossen. Solche Abwägungen sind ganz typisch für Packingprozesse, als Folge stellt seit jeher der Kompromiss zwischen Genauigkeit und Geschwindigkeit ein Dilemma für Forscher im Bereich der Packingoptimierung dar [X. Liu et al., 2015]. Im Fall eines ganzheitlichen Ansatzes kann jedoch davon ausgegangen werden, dass eine gemeinsame Behandlung von kontinuierlichen und diskreten Elementen notwendig ist, da sowohl kontinuierliche (Position und Orientierung, Dimensionierung und Abmaße usw.) als auch diskrete Elemente (Topologievarianten, Anzahl der Komponenten, Verbindungen, Kreuzungen usw.) enthalten sind. In jedem Fall sollte eine einheitliche geometrische Parametrisierung sowohl diskreter als auch kontinuierlicher Variablen durchgeführt werden können, um verbesserte Problemformulierungen zu ermöglichen [Peddada et al., 2021b].

Typischerweise sind Layout- und Packingprobleme bis zum jetzigen Zeitpunkt aufgrund ihrer innewohnenden Komplexität weder durch menschliche Kognition noch durch computergestützte Suche allumfassend zu bewältigen [Peddada et al., 2021b]. Die Grenzen der bestehenden Methoden liegen darin, dass der Zeitaufwand zur Synthese und Analyse von möglichen Lösungen so groß ist, dass es auf Systemebene nicht gelingt, alle Möglichkeiten durchzuspielen und lediglich suboptimale Varianten bei der Entscheidungsfindung berücksichtigt werden. Folglich kann davon ausgegangen werden, dass Layout- und Packingprobleme in Optimierungsprobleme mit zu optimierenden Entwurfsvariablen abstrahiert werden können. Der Entwurfsraum ist dabei immens und anspruchsvoll zu navigieren, da es je nach Entwurfsparametern nahezu unendlich viele Gestaltungsmöglichkeiten geben kann. Unter diesen Voraussetzungen ist es unerlässlich, geeignete Suchstrategien für die Optimierung zu entwickeln, welche den Gestaltungsraum gründlich und effizient abdecken können. Auch [Peddada et al., 2021b] erkennt diese Tatsache und schreibt „Es besteht die Notwendigkeit, durch den erfolgreichen Einsatz leistungsstarker Methoden der Entwurfsautomatisierung, wie z. B. der Entwurfsoptimierung,

rasche Fortschritte [...] zu erzielen, um sich in Entwurfsräumen orientieren zu können, die zu komplex sind, um ausschließlich von menschlichen Fachkenntnissen vollständig erfasst zu werden[1]“. Vor allem die Kombination von kontinuierlichen und diskreten Freiheitsgraden stellt mögliche Optimierungsverfahren dabei vor große Herausforderungen. Deterministische globale Verfahren sind hierbei in der Lage multimodale Problemen zu lösen, wobei die gefundene Lösung innerhalb einer vorgegebenen Toleranz garantiert die beste ist. Ist die Rechenzeit begrenzt, liefern diese potenziell nur ein lokales Optimum und damit eine suboptimale Lösung. Soll tatsächlich eine globales Optimum gefunden werden, können die Rechenzeiten bei komplexen Problemen zu inakzeptablen Rechenzeiten führen. Lokale wie beispielsweise gradientenbasierte Algorithmen schaffen im Punkt Rechenzeit Abhilfe. Sie können die für die Konvergenz zu einer optimalen Lösung erforderliche Rechenzeit erheblich reduzieren, indem sie die Suche auf vielversprechende Richtungen beschränken. Da sie aber nicht in der Lage sind, topologische Entscheidungen abzubilden, muss hierfür häufig in einem Vorverarbeitungsschritte ein möglicher topologischer Ausgangsentwurf mit einem anderen Verfahren berechnet werden. Die Lösung der Optimierung ist dann außerdem lediglich das nächstgelegene lokale Optimum zu dem Ausgangsentwurf. Für eine zufriedenstellende Lösung sind also günstige Ausgangssituationen notwendig. Um aus minderwertigen lokalen Optima zu entkommen, ist es oft erforderlich, dass ein Algorithmus ein gewisses Maß an Zufälligkeit aufweist [Cagan et al., 2002]. Diese Eigenschaft bieten die sogenannten stochastischen Verfahren. In den meisten Fällen werden diese in Form von Metaheuristiken umgesetzt, aber auch globale theoretisch deterministische Optimierer stellen immer häufiger stochastische Methoden für das schnellere Auffinden von suboptimalen Lösungen zur Verfügung (wodurch allerdings im Gegenzug der Determinismus verloren geht). Die Quantität der Zufälligkeit ist in jedem Fall von entscheidender Bedeutung, da die Zufälligkeit die Konvergenz der Suche reduziert und im schlechtesten Fall auch wieder zu inakzeptabel hohen Rechenzeiten führen kann.

Abbildung 2.1 fasst die fünf beschriebenen Teilaspekte zusammen. In Übereinstimmung mit der Darstellung veranschaulicht [Cagan et al., 2002] die in seinen Augen notwendigen Hauptbestandteile zur Lösung eines allgemeinen Layoutproblems. Dazu gehören ebenfalls die geometrische Darstellungen der Komponenten, die Definition von Entwurfsvariablen, einschließlich Lage und Ausrichtung, die Definition von Beschränkungen, die Definition von Zielen und ein geeignetes Optimierungsverfahren. [Cagan et al., 2002] stellt außerdem fest, dass die Definition der Ziele und Einschränkungen von der konkreten Problemstellung abhängen, während der Optimierungsalgorithmus und die geometrische Darstellung sowie die daraus resultierende Interferenzbewertung problemunabhängig seien und daher im Mittelpunkt eines generischen Layouttools stehen sollten. Aus Sicht der Autorin der vorliegenden Arbeit stellt diese Annahme eine ausgezeichnete Basis für die Modularisierung eines Packingframeworks dar, beinhaltet jedoch eine beträchtliche Simplifizierung, die nicht verleugnet werden darf und daher im Folgenden diskutiert wird.

Auch wenn die geometrische Repräsentation und die Optimierung in einem Packingframework zur Verfügung gestellt werden müssen, sind diese nicht im Allgemeinen problemunabhängig. Die besondere Herausforderung beim Packing besteht gerade in der Interdependenz der einzelnen Domänen. Diese sind durchaus gekoppelt und müssen als eine konsistente Optimierungsaufgabe formuliert werden können. Die Eigenschaften der explizit modellierten mathematischen Formulierungen aus allen definierten Nebenbedingungen sowie die implizit enthaltenen mathematischen Gleichungen in den Geometriemodellen und deren Interferenzüberprüfung bestim-

[1] „Hence, there exists an emergent need [for] rapid progress in design capability [...] by successfully leveraging powerful design automation methods, such as design optimization, to help navigate design spaces that are too complex for expert human cognition alone“; Übers. d. Verf.

men dabei den Charakter des Problems, zu welchem ein geeigneter Optimierungsalgorithmus

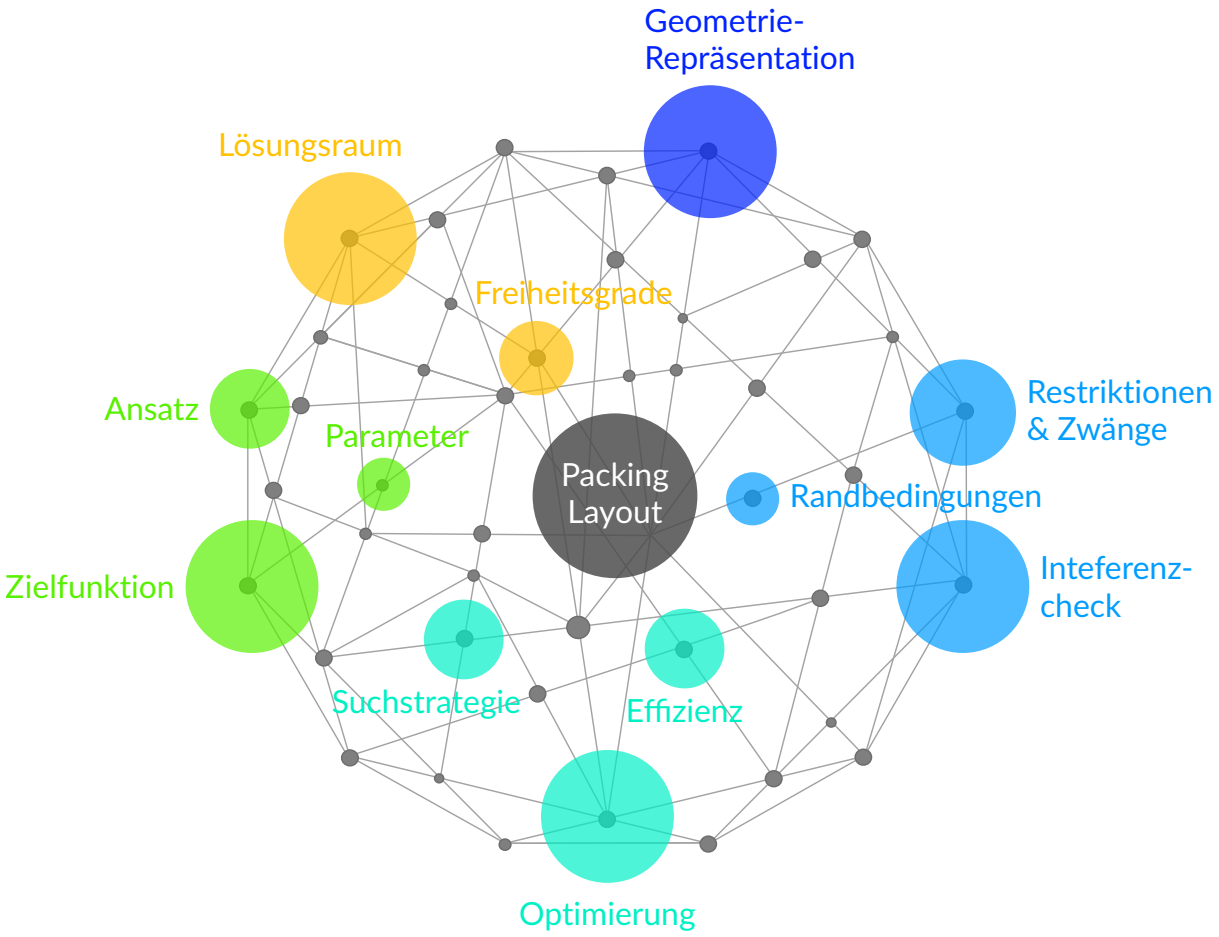


Abbildung 2.1: Die Hauptaspekte zur Lösung eines allgemeinen Layoutproblems

ausgewählt werden muss. Es ist zu beachten, dass sich der Typ des Gesamtproblems immer aus dem komplexesten Anteil ergibt. Wenn zum Beispiel $(n-1)$ mathematische Formulierungen linear sind und eine Formulierung von quadratischer Natur ist, muss das gesamte Problem als quadratisch behandelt werden. Dann muss ein Optimierungsalgorithmus zur Verfügung stehen (oder entwickelt werden), der quadratische Probleme lösen kann. Steht dieser nicht zur Verfügung, muss die Problemdefinition in der Weise umformuliert werden, dass sie von dem vorhandenen Optimierungsverfahren gelöst werden kann. Dabei nimmt mit steigendem Grad der mathematischen Gleichungen die Zahl der geeigneten Optimierungsalgorithmen ab, da die Lösung des Problems komplexer wird. Häufig sind Lösungsalgorithmen auch für spezielle Problemklassen ausgelegt, in denen sie effizient rechnen, weisen allerdings im Gegenzug Schwächen bei anderen Problemklassen bezüglich Effizienz oder Genauigkeit auf. Eine Umgehung dieser Problematik ist die Aufspaltung des Gesamtproblems in einzelne Teilprobleme, welche von spezialisierten Algorithmen oder Optimierern aufeinanderfolgend und iterativ gelöst werden. Bei Unlösbarkeit eines Teilproblems muss dabei zu dem vorherigen Subproblem zurückgesprungen werden, um die Eingangsparameter für das zunächst unlösbare Teilproblem zu verändern. Voraussetzung für eine effektive Aufspaltung ist allerdings das Vorhandensein schwach gekoppelter Teilprobleme innerhalb des Gesamtproblems, was nicht immer der Fall sein muss. Es ergibt sich also eine gegenseitige Abhängigkeit. Die Formulierung des Problems beeinflusst die Wahl des Optimierungsalgorithmus, die Wahl des Optimierungsalgorithmus beeinflusst wiederum die

Formulierung. Diese gegenseitige Abhängigkeit betrifft ebenso die Geometriepräsentation wie auch die Interferenzmodelle. Es bleibt also letztlich, dass ein Optimierungsschema hinreichend flexibel sein muss, um den vorliegenden Problemfall mit all seinen abhängigen Teilbereichen konsistent abbilden und optimieren zu können. Daneben muss für die Behandlung potenziell iterativ zu lösender Teilprobleme die Möglichkeit für die Umsetzung von Multi-Loop-Systemen bestehen. Aber nicht nur die mögliche Aufteilung in Teilprobleme erfordert Schleifen im Lösungsprozess. Allein die schier unendlich große Lösungsvielfalt bei der Packingoptimierung und die damit verbundene Rechenzeit machen es im Falle von System-of-Systems Problemen notwendig schrittweise Optimierungen durchzuführen, wobei ganze Systeme als individuelle Komponenten betrachtet werden können.

Alle bis hierher genannten Herausforderungen müssen zunächst von einem automatisierten Packing- und Layoutprozess gehandhabt werden können. Im Sinne der angestrebten Entwicklung eines ganzheitlichen automatisierten Packingframeworks ergibt sich nun zwingend eine weitere zentrale Forderung - die Integrationsfähigkeit des Packingframeworks in den Gesamtentwurf. Wie in Kapitel 2 beschrieben, muss ein ganzheitlicher Ansatz neben der bloßen Positionierung und Orientierung von einzelnen Komponenten sowohl deren Dimensionierung als auch deren Schnittstellen zueinander berücksichtigen. Nun sind Packing- und Routing-Probleme bereits einzeln NP-schwer, wodurch die Lösung des kombinierten Problems inklusive der geometrischen Kopplungen und physikalischen Interaktionen zwischen den Systemkomponenten besonders schwierig wird. Die Probleme müssen jedoch gleichzeitig gelöst werden, um tatsächlich systemoptimale Entwürfe zu erhalten, ein sequentielles Verfahren wie „packen - dann routen“ oder umgekehrt kann die Wechselbeziehung zwischen allen Entwurfsentscheidungen nicht vollständig erfassen [Peddada et al., 2021b]. Zwar sind in der Literatur viele Ansätze und Modelle bekannt, welche sich mit den einzelnen Problemelementen wie Elementauslegung, Positionierung und Vernetzung befassen, diese sind allerdings auf den jeweiligen Fall zugeschnitten und können nicht einfach in eine umfassende Darstellung übernommen werden. Für einen ganzheitlichen Ansatz besteht die Notwendigkeit einer einheitlichen und konsistenten Repräsentation aller Problemelemente, welche in einer umfassenden Modellierung kompatibel sind und die verschiedenen Problemeigenschaften eines ganzheitlichen Packings mit ausreichender Exaktheit erfassen können [Peddada et al., 2021b]. Dies setzt zum einen eine ausreichend hohe Flexibilität des Frameworks voraus, welche es erlaubt Modelle aus verschiedenen Domänen miteinander interagieren zu lassen und die Domänenunabhängigkeit, sodass alle Domänen gleicherweise abgebildet werden können. Die Einbeziehung verschiedener Domänen führt zu einem weiteren Punkt. Packingaufgaben finden innerhalb eines systemtechnischen Bereiches statt, aber auch an den Schnittstellen zwischen verschiedenen systemtechnischen Anwendungen. Um Entwurfswissen zwischen verschiedenen Gruppierungen von Domänenexperten und -expertinnen effektiv zu kommunizieren, bedarf es also außerdem einer gemeinsamen Terminologie [Peddada et al., 2021b], oder in anderen Worten, einer gemeinsamen Entwurfssprache. Daneben sollte eine Plattform für die Automatisierung von Packingprozessen durchaus auch in der Lage sein, die menschliche Perspektive in allen Schritten des Problemlösungsprozesses zu berücksichtigen. Insbesondere praktizierende Ingenieure und Ingenieurinnen aus der Industrie, die über große Erfahrung im Umgang mit komplexen Systemen verfügen, besitzen wertvolles Entwurfswissen, das bei dem Aufbau von automatisierten Packingsystemen wiederverwendet werden muss [Peddada et al., 2021b]. Sind Menschen in den Prozess integriert, bleibt die Frage nach einer geeigneten Möglichkeit der Visualisierung der Entwurfsergebnisse vor allem im Sinne eines optimalen Punktes im Entwurfsraum. Es müssen also für Visualisierungszwecke geeignete Werkzeuge bereitgestellt werden, welche den Anwendern ermöglichen, die Ergebnisse zu interpretieren und am Ende möglicherweise nachträglich manuell und subjektiv zu bewerten.

2.2 Wissenschaftliche Motivation und Lösungsansatz

Die Essenz aus der vorangegangenen Diskussion ist, dass die größte Hürde bei der Schaffung eines automatisierten Layout- und Packingsprozesses bis heute die fehlende ganzheitliche Behandlung der packingspezifischen Teilprobleme ist. Auch [Peddada et al., 2021b] erkennen diese Tatsache an: „The main limitation with methods used in component packing, interconnect routing, and physics-based topology optimization is that existing efforts address these problems separately, rather than in a combined manner that accounts for inherent coupling.“, frei übersetzt „Die größte Einschränkung bei den Methoden für das Packing von Bauteilen, das Routing von Verbindungen und die physikbasierte Topologieoptimierung besteht darin, dass die bestehenden Verfahren diese Probleme getrennt behandeln und nicht in einer kombinierten Weise, die die inhärente Kopplung berücksichtigt“. Es wird also eine Plattform für das automatisierte Layouting und Packing von vernetzten Systemen unter Berücksichtigung der Wechselwirkungen aus verschiedenen Domänen benötigt, welche eine konsistente Repräsentation und Verarbeitung aller Teilprobleme des ganzheitlichen Packings auf Systemebene ermöglicht. Aus den im vorangegangenen Abschnitt beschriebenen Kernthemen lassen sich die benötigten Forderungen an ein solches Framework nach und nach ableiten.

Zu Beginn sei die Modellierung von den in das Packingproblem eingehenden Anforderungen und Randbedingung genannt. Ein Framework muss Bibliotheken zur Verfügung stellen, welche die Formulierung von Randbedingungen unterstützen. Dabei muss eindeutig spezifiziert werden können, welcher Art die Entwurfsparameter sind, wie sie untereinander abhängen und nicht zuletzt in welchem Wertebereich sie vom Packingalgorithmus variiert werden dürfen. Neben der Eingabe der Entwurfsvariablen muss auch die Spezifikation von physikalischen Bedingungen und Zielen assistiert und verständlich repräsentiert werden können. Zwei weitere genannte Aspekte sind eine geeignete Geometrierepräsentation und ein damit eng gekoppelter Interferenzmechanismus. Bereits die Notwendigkeit einer Geometrierepräsentation führt auf die Forderung, dass innerhalb eines Packingframeworks Werkzeuge zur Verfügung gestellt werden müssen, um geometrische Informationen zu verarbeiten, zu speichern und umzuwandeln. Ferner wird darauf aufmerksam gemacht, dass die Art der Geometriedarstellung und der daraus resultierende Interferenzcheck ausschlaggebend für die Rechenzeiten einer Packingprozedur sein kann. Im Sinne einer effizienten Berechnung ist es daher zusätzlich wünschenswert, dass geometrische Repräsentationen durch sog. Geometrieapproximationen vereinfacht werden.

Weiterhin wird im vorangegangenen Abschnitt auf die Interdependenz der einzelnen Teilaufgaben innerhalb eines Packingprozesses eingegangen. Dabei wird erörtert, dass ein Optimierungsschema hinreichend flexibel sein muss, um ein Packingproblem konsistent abbilden und optimieren zu können. Übertragen auf die Randbedingungen eines Packingframeworks führt dies zu der Erkenntnis, dass es mit hoher Wahrscheinlichkeit nicht möglich ist, jedes Packingproblem derart zu abstrahieren, dass eine einzige Packingstrategie dieses effektiv lösen kann. Es ist folglich unabdingbar, dass ein Packingframework nicht aus einer einzigen starr implementierten Vorgehensweise besteht, sondern viel mehr die Kombination von möglichen vorhandenen Packingstrategien ermöglicht und damit die Modellierung eines problemabhängigen Packings in allen seinen Facetten und Teilbereichen unterstützt. Zum einen bedeutet das, dass die Repräsentation eines Packingproblems und die Modellbildung der Teilmodelle flexibel veränderbar sein muss. Wird bei einem Packingproblem von einem Optimierungsproblem ausgegangen, bedeutet das außerdem, dass unterschiedliche Optimierungsmethoden zur Verfügung gestellt werden müssen sowie Transformationsprozesse, welche die eingehenden Randbedingungen für diese verarbeitbar machen. Hinsichtlich der Optimierung ist außerdem zu berücksichtigen, dass die gegebene Hardware eine entscheidende Rolle in der Rechenleistung spielt und gegebenen-

falls die Optimierung an diese angepasst werden muss. Diese Erkenntnis stimmt mit der Aussage von [Peddada et al., 2022] überein: „Effektive Methoden unterstützen Anpassungen, die leicht vorgenommen werden können, wenn sich die Systemanforderungen im Laufe der Zeit ändern. Ein wichtiger potenzieller Vorteil solcher Methoden ist die Verringerung der Entwurfszeit und der Ressourcen, die zur Lösung [...] erforderlich sind[2]“. Nun kann im Allgemeinen davon ausgegangen werden, dass die Rechenzeit zur Lösung eines Packingproblems von der Anzahl der zu optimierenden Kriterien abhängt. Um diese in akzeptablen Bereichen zu halten, sollte weiterhin auch der Grad der Optimierung flexibel anpassbar sein. Zum einen sollte hierfür zwischen Optimierungskriterien variiert werden können, zum anderen wäre es wünschenswert, flexibel zwischen Optimierung und Entwurfsraumexploration wechseln zu können. Dies wird auch dadurch begründet, dass die Formalisierung der Zielfunktion nicht immer möglich ist, sodass im Nachhinein doch subjektive Einschätzungen des Ingenieurs zur Auswahl einer Variante führen sollen. In Übereinstimmung damit fordert auch [Fadel und Wiecek, 2015], dass „aus den berechneten Konfigurationen [...] Designer eine bevorzugte Konfiguration auswählen [können], die zusätzliche, bei der Optimierung nicht berücksichtigte Kriterien erfüllt[3]“.

Wie zu Beginn erläutert, liegt der Kernpunkt bei der Entwicklung eines Frameworks für automatisiertes Packing allerdings nicht (ausschließlich) in der Beantwortung der einzelnen Teilfragestellungen des Packingprozesses selbst, sondern vor allem im Aufbau einer konsistenten, interoperablen und effizienten Datenstruktur, welche notwendig ist, um den Packingprozess in seiner Gesamtheit zu digitalisieren und zu automatisieren. Dies wurde auch in [Yakovlev et al., 2018] erkannt: „Es bleibt die Frage nach der Schaffung einer optimalen Datenstruktur in einem bestimmten Format unter Berücksichtigung der Klassenhierarchie, der Kriterien der Vollständigkeit, etc.[4]“. Zusammenfassend können die diskutierten Kernthemen zu den folgenden notwendigen Elementen und Forderungen an ein automatisiertes Packingframework kondensiert werden:

- Konsistente Wissensrepräsentation
- Flexibilität
- Erweiterbarkeit und Anpassungsfähigkeit
- Modularität
- Integrationsfähigkeit im Gesamtentwurf
- Domänenunabhängigkeit
- Visualisierungsfähigkeit

Ähnliche Forderungen finden sich auch in der Veröffentlichung von [Yakovlev et al., 2018] wieder, in der zusammenfassend die „Schaffung hochintelligenter Informationstechnologien für die Synthese komplexer Systemkonfigurationen auf der Grundlage konstruktiver Ansätze der Informationsverarbeitung, mathematischer und computergestützter Modellierung, moderner

[2] „Effective methods will support adjustments that can be made easily as the system requirements change over time. An important potential benefit of realizing such methods is the reduction in design time and resources required to solve [...] problems [...].“; Übers. d. Verf.

[3] „From among the computed configurations designers can select a preferred configuration that satisfies additional criteria not considered in the optimization.“; Übers. d. Verf.

[4] „The open question remains the creation of an optimal data structure in a certain format taking into account the hierarchy of classes, criteria of completeness, etc.“; Übers. d. Verf.

Methoden der Optimierung und Visualisierung von Konfigurationen, die an spezifische technologische Prozesse angepasst sind[5]“ gefordert wird. Es bleibt die Frage nach einer geeigneten Repräsentationsform, welche alle genannten Bedingungen erfüllt.

Weniger mit speziellem Fokus auf Layout- oder Packingproblemen und viel mehr im allgemeinen Kontext einer umfassenden Entwurfsmethodik schlägt [Rudolph, 2002] die sogenannten graphenbasierten Entwurfssprachen als Framework zur ganzheitlichen Erfassung und Repräsentation des Ingenieurentwurfs vor. Ähnlich den genannten Erkenntnissen erkennt der Autor: „Die Hemmnisse bzw. Ursachen für das bisherige Scheitern der Erstellung einer für eine allgemeine Entwurfstheorie tauglichen konzeptionellen Modellvorstellung [...] liegen in der Unterschiedlichkeit und Unverträglichkeit der im Ingenieurentwurf gleichzeitig anzutreffenden Modellbeschreibungen begründet“, welche in „topologisch und parametrisch getrennte Lösungsklassen“ eingeteilt werden und deren Informationen in „unterschiedlichen verbalen, symbolischen, analytischen und numerischen Teildarstellungen“ repräsentiert sind. Die zur Lösung des Problems vorgeschlagenen graphenbasierten Entwurfssprachen ermöglichen es, „die für den Ingenieurentwurf relevanten Teilbeschreibungen in einem einheitlichen Datenformat“ dimensionshomogen und konsistent zusammenzufassen [Rudolph, 2002]. Mit den graphenbasierten Entwurfssprachen wird außerdem ein Verfahren vorgeschlagen, das in Übereinstimmung mit der Aussage „Solche Ansätze basieren auf modernen Mitteln der objektorientierten Programmierung zur Verarbeitung und Transformation von Datenstrukturen, Parallel-Computing-Technologien [und] 3D-Simulationstechnologien zur Visualisierung synthetisierter Konfigurationen[6]“ von [Yakovlev et al., 2018], auf den Grundsätzen der objektorientierten Programmierung basiert.

Der Wunsch nach einer informationstechnischen Struktur zur Unterstützung von mehrdimensionalen und multikriteriellen Layout- und Packingprozessen im Systementwurf in Kombination mit den Konzepten von [Rudolph, 2002] führen abschließend zu der Aufgabenstellung der vorliegenden Arbeit, in kurzen Worten, ein Framework für automatisiertes Packing mithilfe von graphenbasierten Entwurfssprachen zu entwickeln.

[5] „On the other hand, the problem is the creation of high-intelligence information technologies for the synthesis of complex systems' configurations based on constructive approaches to information processing, mathematical and computer modeling, modern methods of optimization and visualization of configurations adapted to specific technological processes.“; Übers. d. Verf.

[6] „Such studies are based on modern means of object-oriented programming for processing and transformation of data structures, parallel computing technologies, 3D simulation technology for visualization of synthesized configurations.“; Übers. d. Verf.

KAPITEL 3

Stand der Technik im Bereich des automatisierten Packings

Die Optimierung eines räumlichen Packings von vernetzten Systemen spielt in nahezu allen Bereichen der Systementwicklung eine wichtige Rolle. Die in der vorliegenden Arbeit als *Packing* bezeichnete Aufgabenstellung fällt in der Literatur unter viele weitere Überschriften, wie *Packaging*, *Komponentenanordnung*, *-layout* oder *-konfiguration*, *räumliche Platzierung*, *Positionierung* und *Orientierung* oder *Unterbringung*, und wird ebenso unter den englischen Fachbegriffen *container stuffing*, *pallet loading*, *nesting*, *cutting*, *covering* oder *partitioning* aufgeführt. Dabei erstrecken sich Packingaufgaben in der Literatur von der bloßen translatorischen Positionierung, über die zusätzliche rotatorische Orientierung bis hin zu der darüber hinausgehenden Veränderung der Gestalt der Einzelelemente und berücksichtigen fast immer die Kollisionsfreiheit von Elementen bis hin zu einer Reihe an geometrischen Nebenbedingungen wie beispielsweise Abstände oder Kontakt, und beziehen in seltenen Fällen auch physikalische Randbedingungen aus den Bereichen Mechanik, Thermodynamik, Hydraulik, Optik und vielen weiteren mit ein. Über die letzten Jahrzehnte haben solche Packingprobleme bereits in zahlreichen Industriebereichen Anwendung gefunden, wie beispielsweise im Leiterplattenlayout, in der Logistik, in der Transportindustrie, bei der Gepäck- und Containerbeladung ([Bortfeldt und Wäscher, 2013], [Paquay et al., 2014]), in der Bekleidungsindustrie und Möbelherstellung, beim Glas- oder Metallschneiden, in der Blechbearbeitung sowie in der Fertigungsindustrie [Drira et al., 2007], in der additive Fertigung ([Edelkamp und Wichern, 2015], [Lutters et al., 2012] und [Araújo et al., 2019]), in Architektur und Bauwesen, sowie im Maschinenbau, wie z.B. in der Luft- und Raumfahrt ([Teng et al., 2001], [Zhi-Guo und Hong-Fei, 2003], [Giorgio Fasano, 2014], [Giorgio Fasano et al., 2015]) oder in der Automobilindustrie, wie z.B. bei der Auslegung von Transportfahrzeugen ([Yi et al., 2008]), beim Packing von Motorraumkomponenten ([Reichel, 2006], [Dong et al., 2011], [Dandurand et al., 2014]) und beim Packing von Gepäckstücken im Kofferraum ([Tiwari et al., 2008], [Tiwari et al., 2010]) um nur eine Auswahl an Domänen und Disziplinen zu nennen.

Das vorliegende Kapitel soll eine Übersicht über den Stand der Technik im automatisierten Packing geben und gleichzeitig das notwendige Hintergrundwissen zum Verständnis der folgenden Kapitel vermitteln. Zunächst werden in *3.1 Klassifikation in der Literatur* mögliche Varianten der Einteilung von Packingproblemen vorgestellt. Nachfolgend wird in *3.2 Randbedingungen, Ziele und Freiheitsgrade* darauf eingegangen, wie mögliche Randbedingungen und Ziele für den Packingprozess dargestellt werden können und in welcher Form sie bereits in der Literatur behandelt wurden. Da ein weiterer zentraler Bestandteil bei der Modellierung von

Packingproblemen die Repräsentation der Geometrie ist, wird dieser Punkt in *3.3 Repräsentation von Geometrie und Interferenzcheck* genauer vorgestellt. Für die Lösung von Packingaufgaben werden in der Regel Optimierungsverfahren eingesetzt. Hier existiert in der Literatur eine Vielzahl möglicher Herangehensweisen, welche in *3.4 Optimierungsverfahren für Packingprobleme* zusammengefasst und besprochen werden. Da das Hauptaugenmerk der vorliegenden Arbeit auf einer ganzheitlichen Behandlung von Packingprozessen liegt, werden bereits veröffentlichte Ansätze mit einem ähnlichen Ziel in *3.5 Ganzheitliche Ansätze und die Einbettung im Gesamtentwurf* separat genauer beleuchtet. Abschließend wird das Kapitel mit *3.6 Graphenbasierte Entwurfssprachen* abgerundet, wo die Graphenbasierten Entwurfssprachen als Plattform für die Umsetzung eingeführt werden.

3.1 Klassifikation in der Literatur

Die in der Literatur verwendeten Klassifizierungen und Kategorien sind inhomogen und werden nach verschiedenen Kriterien festgelegt. Nach [Wäscher et al., 2007] wird die Klassifizierung in der Regel durch folgende Faktoren bestimmt: die Dimensionalität des Raums (1D, 2D, 3D oder nD), die Formkomplexität der Objekte, welche regelmäßige Objekte (Rechtecke, Kreise, Kästchen, Zylinder, Kugeln usw.) oder unregelmäßige Objekte (auch als nicht-regulär bezeichnet) beinhalten kann, die Zusammenstellung der Objekte, also ob es sich ausschließlich um identische oder heterogene Geometrien handelt, die Diskretisierung des Lösungsraums, mit der Unterteilung in eine orthogonale Platzierung, nicht-orthogonale Platzierung, ein Raster, ein Gitter usw., und die Frage, ob es sich um ein Auswahlproblem (welches auch als *Output-Maximierung* bezeichnet wird) oder ein räumliches Positionierungsproblem (welches auch als *Input-Minimierung* bezeichnet wird) handelt.

[Fadel und Wiecek, 2015] dagegen gruppieren Packingverfahren vergleichsweise übersichtlich in *kompaktes* und *nichtkompaktes* Packing. Ziel des ersteren ist Freiformobjekte in einem beliebig geformten Gehäuse so zu platzieren, dass die Anzahl der in das Gehäuse passenden Objekte oder deren Volumen maximiert wird. Das einzige Kriterium für die Zielfunktion des *kompakten* Packings ist die Dichte, die durch eine möglichst enge Platzierung von Systemkomponenten in einem Gehäuse maximiert werden soll. Das *nichtkompakte* Packing dagegen, von [Fadel und Wiecek, 2015] auch bezeichnet als Konfigurationslayout, unterscheidet sich hauptsächlich darin, dass die dichteste Packung nur eins von vielen weiteren Kriterien zur Bewertung der Systemleistung ist. Das Vorhandensein mehrerer Bewertungsparameter führt auf ein multikriterielles Optimierungsproblem, das mehrere Zielfunktionen berücksichtigen muss. Es kann also geschlossen werden, dass die Klassifikation nach [Fadel und Wiecek, 2015] im Wesentlichen auf der Anzahl der Zielkriterien des Packingproblems basiert.

In [Yakovlev und Kartashov, 2018] wird eine geometriebasierte Klassifizierung unter dem Gesichtspunkt der Systemanalyse vorgeschlagen. Darin wird der Konfigurationsraum, also der Lösungsraum aller möglichen Konfigurationen, zur Formalisierung von sog. Syntheseproblemen verwendet. Für die Analyse werden die Variablen in metrische Parameter und Positionierungsparameter eingeteilt, die Nebenbedingungen in geometrische und physikalisch-mechanische. In der Klassifikation wird davon ausgegangen, dass im Allgemeinen alle Freiheitsgrade des Packings durch die Definition von Nebenbedingungen verschiedenen Restriktionen unterliegen, die ihren zulässigen Wertebereich eingrenzen. Basierend darauf bestimmt laut [Yakovlev und Kartashov, 2018] die Wahl der Variablen und der Nebenbedingungen die entsprechende Klasse von Packingkonfigurationen. Die vorgestellte Klassifikation beinhaltet die Klassen *Layout Configuration* (bei Vorhandensein von Beschränkungen bezüglich der zulässigen Mindest-

und Maximalabstände zwischen Objekten), *Balanced Packing Configuration* (wenn Objekte Festkörper mit gegebenen Massen sind und gleichmäßig verteilt werden sollen), *Covering Configuration* (wenn geometrische Objekte, als überdeckende Objekte bezeichnet, eingebettet werden in ein weiteres geometrisches Objekt, bezeichnet als Abdeckbereich), sowie *Euclidean Combinatorial Configuration* (wenn Platzierungsparameter und metrische Parameter der Objekte diskrete Werte annehmen) [Pichugina und Yakovlev, 2020]. Die Einteilung nach [Yakovlev und Kartashov, 2018] basiert also auf der Kombination der verwendeten Freiheitsgrade und der geometrischen Nebenbedingungen.

[Peddada et al., 2021b] beziehen in einer Unterteilung neben den reinen geometrischen auch die physikalischen Eigenschaften des Systems mit ein. Hier wird festgestellt, dass sich die Problemvarianten gemeinhin durch die jeweilige Definition ihrer geometrischen und physikalischen Entwurfszwänge sowie der Zielfunktion unterscheiden. Weiterhin nehmen [Peddada et al., 2021b] eine Kategorisierung auf Basis der verwendeten Lösungsalgorithmen vor, wobei eine (in den Augen der Autorin der vorliegenden Arbeit unvollständige) Einteilung in vier Kategorien stattfindet: Packing mithilfe genetischer Algorithmen, heuristischer Methoden, gradientenbasierter Algorithmen und Simulated-Annealing-Methoden.

3.2 Randbedingungen, Ziele und Freiheitsgrade

So unterschiedlich die Anwendungsbereiche von Packingproblemen sind, so unterschiedlich sind auch die in der Literatur vorgeschlagenen und behandelten problemspezifischen Randbedingungen, Ziele und Freiheitsgrade. Im Folgenden soll ein kurzer Überblick über eine Auswahl an Kriterien gegeben werden, welche in der Literatur bereits zur Repräsentation von Nebenbedingungen und Zielen bei Packingprozessen berücksichtigt wurden. Der Leser sei an dieser Stelle darauf hingewiesen, dass dieser Überblick nur dem Einblick in die Vielseitigkeit von Packingproblemen dient und keinerlei Anspruch auf Vollständigkeit hat, welcher den Rahmen der vorliegenden Arbeit sprengen würde.

3.2.1 Kriterien für Nebenbedingungen und Ziele

Kompaktheit Die Kompaktheit eines Systems wird häufig anhand des Volumens eines Hüllquaders oder anhand des Volumens der konvexen Hülle des Systems bemessen. [Jessee et al., 2020] und [Peddada et al., 2021a] beispielsweise bestimmen am Beispiel eines fiktiven Kühlsystems für die Leistungselektronik eines unbemannten Flugzeugs optimale Packing- und Routing-Layouts und werten dabei das Volumen des Hüllquaders aus, welcher alle Systemkomponenten umgibt. Auch [Grignon und Fadel, 2004] bedienen sich der Kompaktheit in ihrer Arbeit, in der sie eine Methodik zur Optimierung von Systemeigenschaften komplexer mechanischer Baugruppen durch die Anordnung ihrer Komponenten vorstellen. Dort wird die Kompaktheit allerdings durch eine andere Metrik ersetzt, nämlich das Trägheitsmoment. Dies ist darauf zurückzuführen, dass nach Meinung der Autoren Ansätze mit Hüllkörpern zwar durchaus mathematisch sinnvoll sind, aber zu irreführenden Ergebnissen im Entwurf führen können, da verschiedene Konfigurationen desselben Volumens für den Konstrukteur tatsächlich von unterschiedlichem Nutzen sind. Der Ansatz über das Trägheitsmoment führt dazu, dass sich Komponenten aneinander annähern, ohne das Volumen eines umgebenden Behälters minimieren zu müssen. Um eine Bewertungsmetrik zu erhalten, wird dabei zunächst das Trägheitsmoment für jede Systemkomponente separat berechnet und das Systemträgheitsmoment kann anschließend aus der Summe aller Komponententrägheitsmoment berechnet und als Maß

für die Kompaktheit verwendet werden. Angewandt wurde die vorgestellte Methode auf zwei technische Testfälle aus der Luft- und Raumfahrt beim Packing von Satellitenkomponenten und der Automobiltechnik beim Packing von Motorraumkomponenten.

Gegenseitige Lage Funktionale Randbedingungen wie Abstände zwischen Elementen oder die gegenseitige Lage lassen sich bei Packingprozessen leicht als geometrische Randbedingungen ausdrücken. Häufig sind diese so geartet, dass sie in die Definition der Optimierungsvariablen eingebettet werden können und keine weiteren Gleichheits- oder Ungleichheitsbedingungen erfordern. In anderen Worten können solche geometrischen Randbedingungen über eine Einschränkung des Wertebereichs der Freiheitsgrade mithilfe einer unteren oder oberen Schranke umgesetzt werden oder über das Vernachlässigen ganzer Freiheitsgrade. Auch [Grignon und Fadel, 2004] nutzen diese Art der Einschränkung für Bauteile im Motorraum, welche gegenseitige Abhängigkeiten hinsichtlich der Lage aufweisen. Sie nutzen dabei die Tatsache, dass die Lage von bestimmten Bauteilen über diese Abhängigkeit von vornherein bekannt ist, sodass sie keine eigenen Freiheitsgrade benötigen und es ausreicht, die Freiheitsgrade lediglich eines der beiden Bauteile einzubeziehen. Auch im Bereich der Architektur spielt die gegenseitige Lage eine wichtige Rolle, beispielsweise für die Adjazenz von Räumen. Viele Layoutproblemstellungen gehen davon aus, dass Räume zu Beginn der Problemlösung bereits mit vordefinierten Adjazenzen eingegeben werden, welche während des Lösungsprozesses erhalten bleiben sollen. In [Rodrigues et al., 2013a] werden die Abstände der Mittelpunkte von adjazenten Räumen ausgewertet und dienen als Maß der Einhaltung dieser Forderung.

Kontakt [Belov et al., 2018] beschäftigen sich mit der Optimierung des Layouts von Prozessanlagen. Für die Erfüllung einer Kontaktanforderung zwischen Prozessanlagen und zugehörigen Rohranschlüssen repräsentieren sie den Rohranschluss zu einer Anlage als eine Hilfsbox, welche sich in genau einer von mehreren vorher spezifizierten Zonen an den Seitenwänden der Anlagenbox befinden muss. Während des Packingprozesses bewirkt die Rotation der Anlage auch die Rotation der Zonen und damit der Hilfsbox selbst, sodass die korrekte gegenseitige Orientierung für den Kontakt sichergestellt wird.

Gewichtsverteilung [Teng et al., 2001] and [Zhi-Guo und Hong-Fei, 2003] beschäftigen sich mit der gleichmäßigen, oder in anderen Worten, ausbalancierten Verteilung von Satellitenkomponenten in einem Satellitengehäuse und optimieren dafür das Trägheitsmoment. Auch für [Grignon und Fadel, 2004] im Kontext des Packings von Komponenten eines Motorraums spielt die Gewichtsverteilung eine wichtige Rolle. In ihrer Arbeit unterteilen die Autoren die Gewichtsverteilung in einen statischen und einen dynamischen Beitrag, wobei das statische Systemgleichgewicht erreicht wird, indem der Systemschwerpunkt auf einen Zielwert optimiert wird, während das dynamische Gleichgewicht durch das Systemträgheitsmoment gemessen wird, das, wie oben erwähnt, gleichzeitig auch als Kompaktheitsmaß dient. Ein weiterer Ansatz, welcher Gleichgewichtsbedingungen berücksichtigt, wurde im Bereich des Fahrzeugsektors für die Auslegung der Fahrzeugkonfigurationen von Armeetransportern von [Yi et al., 2008] vorgeschlagen. Die Autoren stellen in ihrer Arbeit eine Methodologie zur Auffindung einer optimalen Fahrzeugkonfiguration vor, bei der das Ziel darin besteht, die Fahrzeugleistung zu maximieren. Darin interpretieren sie das Gleichgewichtskriterium als Wahrscheinlichkeit für einen Überschlag des Fahrzeugs und repräsentieren und quantifizieren dieses über die Querbeschleunigung des Fahrzeugs, welche während des Packingprozesses mithilfe eines Prognosemodells iterativ berechnet wird.

Wartbarkeit [Grignon und Fadel, 2004] definieren für den Motorraum eines Fahrzeuges, dass unter der Wartbarkeit eines Bauteils seine Zugänglichkeit bzw. die Menge an mechanischer Arbeit verstanden wird, die geleistet werden muss, um auf das Bauteil zuzugreifen und es aus dem System zu entfernen. In ihrem Ansatz ist diese Arbeit proportional zum Gewicht der Komponente und aller weiteren Komponenten, welche entfernt werden müssen, um auf die betreffende Komponente zuzugreifen. Die Wartbarkeit des Gesamtsystems wird dann als Summe der einzelnen Wartbarkeiten aller Komponenten repräsentiert. Vergleichbar mit dem Ansatz wird auch bei der Auslegung der Fahrzeugkonfigurationen von Armeetransportern in [Yi et al., 2008] die Wartungsfreundlichkeit als Zugänglichkeit repräsentiert und mithilfe der Bauteile quantifiziert, die entfernt werden müssen, bevor die betreffende Komponente in einer bestimmten Richtung entfernt werden kann. Hierbei wird die Wartbarkeit jedoch lediglich durch die Anzahl der auszubauenden Komponenten repräsentiert und nicht durch deren Gewicht. [Belov et al., 2018] wiederum modellieren die Wartbarkeit bei ihrem Ansatz zur Optimierung von Anlagenlayouts. Dort stellen einige Anlagen besondere Anforderungen an den Wartungszugang, z. B. müssen sie von oben bzw. unten oder mit einem Lkw zugänglich sein. Ersteres wird durch eine Einschränkung modelliert, die sicherstellt, dass sich kein anderes Equipment über oder unter der jeweiligen Anlage befindet. Letzteres wird im Modell durch einen leeren Raum repräsentiert, der an die Anlage angeschlossen und von der Straße aus zugänglich ist. Dies wird durch zusätzliche Hilfsboxen modelliert, die bestimmte Anforderungen bezüglich Größe und Lage der Anlagenboxen erfüllen. D.h. dass die Forderung nach Wartbarkeit in eine relative Positionierung transformiert wird. Dabei werden zwei Arten von Wartbarkeitsanforderungen modelliert. Im ersten Fall muss der Zugang zu einer bestimmten Seite des Geräts gewährleistet sein. Dies wird über eine starre Befestigung repräsentiert, bei der sich die Hilfsbox an einer bestimmten Position relativ zur zugehörigen Anlage befindet und die gesamte Kombination sich gemeinsam dreht. Der zweite Fall ist die Anforderung, dass der Zugang lediglich zu einer beliebigen Seite des Geräts möglich sein muss. Diese wird über eine drehbare Befestigung modelliert, in welcher die Hilfsbox zusätzlich um die zugehörige Anlage rotiert werden kann.

Thermodynamik [Katragadda et al., 2012] befassen sich ebenfalls mit dem Beispiel des Motorraums und erweitern diesen um thermodynamische Nebenbedingungen. Wegen der vom Motor erzeugten Wärme wird die zusätzliche Bedingung berücksichtigt, dass bestimmte Bauteile in einiger Entfernung von den Wärmequellen platziert werden müssen. Für dieses Problem wird auf jedes Bauteil eine zusätzliche Temperaturvariable angewandt, und es werden Wärmequellen vorgegeben. Die Temperaturverteilung in der Motorhaube wird mithilfe von Strömungssimulationen berechnet. Die Nebenbedingung wird dabei so formuliert, dass die Temperatur kritischer Komponenten wie der Batterie unter einem kritischen Wert gehalten wird. Außerdem wird das zusätzliche Ziel formuliert, den Effektivwert der durchschnittlichen Temperaturen unter der Motorhaube zu minimieren. Auch [Jessee et al., 2020] und [Peddada et al., 2021a] berücksichtigen temperaturabhängige Kriterien. Sie verwenden für technische Systeme den Gesamtwärmeverlust und die maximale Komponententemperatur als Leistungskriterien für die Optimierung von Packing- und Routing-Layouts auf Systemebene. Die thermalen Berechnungen basieren dabei auf der Auswertung von Finite-Element-Modellen.

Hydraulik Neben thermalen Betrachtungen unterstützt das in [Jessee et al., 2020] und [Peddada et al., 2021a] vorgestellte Framework zusätzlich hydraulische physikalische Betriebsbedingungen, wie den hydraulischen Druck und die Durchflussrate von Fluiden, welche mithilfe von Finite-Element-Modellen berechnet werden. Damit stellen sie Möglichkeiten vor, wie hydraulische Randbedingungen in Packingstrategien integriert werden können.

Überlebensfähigkeit Die Überlebensfähigkeit ist vor allem bei der Auslegung von militärischen Systemen von fundamentaler Bedeutung. Bei der Fahrzugkonfiguration von [Yi et al., 2008] wird die Überlebensfähigkeit einer Fahrzeugkomponente als sog. Schutzniveau repräsentiert, welches aus der Überlappung mit anderen Komponenten entlang einer ausgewählten Richtung berechnet wird. Die Modellierung basiert auf der Annahme, dass die umliegenden Komponenten Schutz vor möglichen zerstörerischen Auswirkungen von außen bieten.

Bodenfreiheit Als weitere fahrzeugspezifische Nebenbedingung wird in [Yi et al., 2008] auch die Bodenfreiheit berücksichtigt. Diese spiegelt sich in der vorgeschlagenen Modellierung im maximalen Neigungswinkel wider, den das Fahrzeug überwinden kann.

Ausleuchtung und Sonneneinstrahlung Ein wichtiges Kriterium für die Qualität eines Gebäudegrundrisses ist die Ausleuchtung des Gebäudes oder in direktem Zusammenhang damit die Sonneneinstrahlung. In [Zawidzki und Szklarski, 2020] wird die Tageslichtqualität als eine Funktion des Winkels zwischen der direkten Sonnenstrahlung und der Außenwandnormalen repräsentiert. Um den Tagesdurchschnitt zu berücksichtigen, werden die Ost-, Süd- und Westeinstrahlung miteinander verrechnet. Die verwendete Tageslichtfunktion, deren Winkel durch die geografische Lage bestimmt werden, geht in die Zielfunktion ein, wodurch eine Optimierung der Lage stattfindet. In [Zheng und Ren, 2020] wird die Sonneneinstrahlung über die Ausrichtung der Gebäude und die gegenseitige Beschattung bewertet. Es wird eine Zielfunktion modelliert, welche belohnt wird, wenn die Balkonseite der Gebäude nach Süden ausgerichtet ist. Für die Bewertung der Abschattung wird zur konservativen Berechnung der Schattengrenzen der lokale Sonnenscheinwinkel zur Wintersonnenwende verwendet. Durch die Ermittlung des prozentualen Anteils der Gebäudefläche, die in jeder Tageszeit nicht von Schatten bedeckt ist, kann der Gesamtzustand der Sonneneinstrahlung quantifiziert werden.

Akustik und Geräuschentwicklung Die Geräuschentwicklung spielt in vielen Domänen eine Rolle, von der Fahrzeugentwicklung bis zur Gebäudeplanung. [Zawidzki und Szklarski, 2020] z.B. stellen einen Ansatz für Fragestellungen in der Architektur vor. Darin wird die Geräuschqualität aus skalaren Geräuschmessungen auf dem Standort abgeleitet. Dafür wird ein Geräuschfeld formuliert, das eine lineare Interpolation der vor Ort gemessenen Werte darstellt. Es wird weiterhin angenommen, dass die Richtung der Lärmquelle parallel zu den Gradienten des Feldes verläuft. Die Lärmbelastung wird schließlich unter Berücksichtigung des Winkels zwischen der Außenwandnormalen und der Lärmrichtung berechnet, wobei ihre Amplitude proportional zu den jeweiligen Funktionswerten des Feldes ist. Für eine dimensionslose Bewertung werden die berechneten Werte außerdem so mit dimensionsbehafteten Werten skaliert, dass sich eine Skala zwischen 0.5 und 1.0 ergibt.

Sicht Eine ebenfalls für die Architektur und das Bauwesen wichtige Größe ist die Aussicht aus einem Gebäude. [Zawidzki und Szklarski, 2020] stellen einen Ansatz vor, in dem die Sichtqualität als Funktion in Abhängigkeit von der Position des Betrachters und dessen Blickrichtung repräsentiert ist. Zuvor findet eine Einteilung der Sichtfelder in Oktanten von jeweils 45 Grad statt, um aus einer anschließenden subjektiven Bewertung der Sicht Maßwerte zu extrahieren, aus denen sich durch Interpolation über das gesamte Baugrundstück eine Heatmap zur Auswertung der Funktion ergibt. Mit dieser Methode versuchen die Autoren weiche Kriterien, wie die Ästhetik, als Zielparameter abzubilden. In anderen Anwendungsfällen repräsentieren aus den Fenstern gemessene Abstände zum nächsten Sichthindernis die Qualität der Sicht.

Kollisionsfreiheit Zuletzt sei noch die Kollisionsfreiheit von Komponenten zu nennen, welche in nahezu allen Fällen eine Grundvoraussetzung für das Packing ist und daher in allen der genannten Arbeiten umgesetzt wird. Auf diese wird unter Berücksichtigung der verwendeten Geometrierepräsentationsformen im folgenden Abschnitt genauer eingegangen.

3.2.2 Freiheitsgrade

Durch die Vielseitigkeit von Packingaufgaben können aus dem Begriff *Packing* alleine nicht zwangsläufig die zu berücksichtigenden Freiheitsgrade abgeleitet werden. Viel mehr hängen diese von den spezifischen Problemen ab und sollten nebenbei noch an die Rechenleistung des zur Verfügung stehenden Systems angepasst werden. Im Folgenden soll ein Überblick gegeben werden, welche typischen packingspezifischen Freiheitsgrade in der Literatur existieren und in welcher Form sie repräsentiert werden können.

Kontinuierliche Freiheitsgrade Der traditionelle Fall für Packingaufgaben enthält kontinuierliche, translatorische und rotatorische Freiheitsgrade. Die kontinuierliche Repräsentation von translatorischen Freiheitsgraden führt aus mathematischer Sicht zu einfachen Problemformulierungen, da sie zu glatten Funktionen bei der Auswertung weiterer Nebenbedingungen wie z.B. der Kollision führen. Rotatorische Freiheitsgrade dagegen können bei einer kontinuierlichen Repräsentation schon eine höhere mathematische Komplexität mitbringen, da sie die Auswertung von trigonometrischen Funktionen erfordern. Mit zunehmender Wahrnehmung des Packings als den kompletten Gesamtsystementwurf betreffendes Problem werden neben der Translation oder Rotation immer häufiger auch dimensionierende Skalierungsvariablen als Freiheitsgrade berücksichtigt. Eine Skalierung kann entweder die Größe einer Komponente verändern oder zu einer Streckung oder Stauchung führen, wenn sie in Form von unabhängigen Skalierungen entlang einzelner Raumrichtungen repräsentiert wird. Sollen Bauteilmaße komplexeren Änderungen unterworfen werden, können die Freiheitsgrade nicht mehr so einfach in Form skalierender Variablen dargestellt werden. Solche Fälle werden im weiteren Verlauf der vorliegenden Arbeit in Abschnitt 3.3.3 detailliert vorgestellt.

Diskrete Freiheitsgrade Je nach Komplexität des Gesamtproblems müssen häufig weitere Vereinfachungen eingeführt werden, etwa indem Freiheitsgrade nicht mehr kontinuierlich, sondern satt dessen diskret repräsentiert werden. Im Kontext der Herstellung von Zuschnitten aus zweidimensionaler Rohware verwenden [Cherri et al., 2019] beispielsweise diskrete rotatorische Freiheitsgrade für die Orientierung der Zuschnitte. Aber auch die Repräsentation von topologischen Entscheidungen kann in Form diskreter Freiheitsgrade stattfinden, z.B. über eine ganzzahlige Variable, deren Werte jeweils für eine Variante stehen. [Cherri et al., 2019] nutzen diesen Ansatz für die Auswahl der Zuschnitte, wobei ein Wert der ganzzahligen Variable genau eine Zuschnittgeometrie repräsentiert. Bei der Optimierung von Grundrissen wird ein solcher Ansatz in [Rodrigues et al., 2013a] verfolgt. In der Veröffentlichung werden nicht nur Räume positioniert und dimensioniert, viel mehr sollen darüber hinaus auch die Positionen der Fenster und Ausgänge nach außen berücksichtigt werden. Pro Fenster oder Ausgang wird eine ganzzahlige Variable eingeführt, deren Werte den Außenwänden des Gebäudes zugeordnet sind, sodass festgelegt werden kann, an welcher Außenwand das Fenster oder die Tür platziert werden sollen. Nur die genaue Positionierung entlang der Wand wird als kontinuierlicher Freiheitsgrad betrachtet.

Boolesche Freiheitsgrade Eine weitere Möglichkeit der Diskretisierung ist durch die Verwendung von binären oder booleschen Variablen gegeben. Auch hierzu kann die Arbeit von [Cherri et al., 2019] als Beispiel angeführt werden. In dieser wird eine räumliche Diskretisierung mithilfe eines Punktrasters durchgeführt, entlang dessen Zuschnitte positioniert werden können. Die Repräsentation der translatorischen Freiheitsgrade findet über die Belegung der Punkte des Rasters statt, d.h. für jeden Punkt wird eine boolesche Variable eingeführt, die durch einen Zuschnitt belegt sein kann und damit die Positionierung eines anderen Zuschnittes an dieser Stelle verhindert. Ein weiterer Ansatz ist im Automobilbereich zu finden. In [Yi et al., 2008] beispielsweise soll die Leistung des Fahrzeugs maximiert werden, indem die optimale Platzierung für acht Komponenten gefunden wird. Zur Reduktion der Komplexität wird bei Komponenten mit Rotationsfreiheitsgraden deren Drehwinkel auf ein Vielfaches von 90 Grad beschränkt. Die rotatorische Ausrichtung wird aber nicht von ganzzahligen Variablen repräsentiert, sondern mithilfe zweier binären Variablen, deren kombinatorische Vielfalt genau vier mögliche Drehungen zulässt.

3.3 Repräsentation von Geometrie und Interferenzcheck

Die Repräsentation der Packinggeometrien ist ein entscheidender erster Schritt bei Packalgorithmen. Wenn die geometrische Gestalt einer Komponente regelmäßig ist, sei es prismatisch, kugelförmig, zylindrisch oder Derivate davon, ist die Positionierung und die Sicherstellung der Kollisionsfreiheit und damit der Packingprozess relativ einfach. Ein großer Teil der Literatur basiert auf solchen Formen, mit einer Fülle von Algorithmen und Ansätzen, die sich mit dem Packing solcher Basisgeometrien befassen. Wenn allerdings die geometrische Form der Komponenten oder die Form des Bauraums unregelmäßig ist, dann ist eine detaillierte Darstellung der Geometrien erforderlich. Tatsächlich ist dies bei den meisten mechanischen Bauteilen der Fall, die in realitätsnahen Packingproblemstellungen betrachtet werden müssen. Es ist daher erforderlich, eine geeignete geometrische Repräsentation des Bauraums zu finden, sowie eine geeignete Umhüllung der zu positionierenden Komponenten, welche eine ausreichend hohe Genauigkeit aufweist, aber dennoch die für das Problem irrelevante detaillierte innere Geometrie verbirgt oder kapselt [Fadel und Wiecek, 2015]. Auch für das Packing von komplexen Geometrien wurden in den letzten Jahrzehnten neue Ansätze entwickelt, die im Folgenden vorgestellt werden sollen. Im Sinne einer ganzheitlichen Betrachtung von Packingproblemen und der gewünschten Einbettung in den Kontext des Gesamtentwurfs, ist für die Repräsentation von Packinggeometrien außerdem eine systematische Vorgehensweise und eine dazu passende und eine Automatisierung unterstützende Bibliothek wünschenswert. Dieses Erfordernis ist auch von einem kleineren Teil der Literatur erkannt worden, sodass einige Ideen und zum Teil auch Ansätze für deren Umsetzungen existieren.

3.3.1 Geometrierepräsentation und -vereinfachung

Die exakte Erkennung von Interferenzen kann sehr rechenintensiv sein, insbesondere wenn die Objekte komplexe Formen haben und aus einer großen Anzahl von Untereinheiten bestehen. Um diese Kosten zu minimieren, werden in der Regel Hüllvolumina anstelle der eigentlichen Geometrie auf Überschneidungen getestet, wenn es die Genauigkeitsanforderungen zulassen. Andere Ansätze testen die Hüllgeometrien lediglich bevor die eigentliche Geometrie getestet wird, sodass der Kollisionstest für die tatsächliche Geometrie nur erforderlich ist, wenn der anfängliche Überschneidungstest der Hüllgeometrien positiv ausfällt. Die Kollisionsdetektion läuft dann in mehrstufigen Verfahren ab. Eine Vereinfachung der Geometriediskretisierung ist

aber nicht auf ein einzelnes Hüllvolumen pro Komponente beschränkt, Geometrien können ebenso durch eine ganze Reihe von Hüllvolumina approximiert werden. Dabei ist zu beachten, dass einige Geometrieannäherungen der Ausgangsgeometrie sowie deren Interferenzprüfung mehr Optimierungsvariablen und -beschränkungen benötigen als andere, was dazu führt, dass die Wahl der Geometrierepräsentation eine erhebliche Auswirkung auf die Komplexität des Gesamtproblems hat. Im Folgenden sollen unter dem Gesichtspunkt der Geometrierepräsentation Konzepte aus der Literatur vorgestellt werden, welche bei Packingaufgaben Anwendung finden.

Tessellierte Oberfläche In den meisten Fällen liegen Geometrien für Packingprozesse zunächst als triangulierte Oberflächennetze vor. Im einfachsten Fall können diese direkt weiterverarbeitet werden, sodass keine zusätzlichen Vereinfachungsprozesse notwendig sind. Der Ansatz von [Grignon und Fadel, 2004] berücksichtigt jede beliebige Bauteilform (einschließlich nicht-konvexer, hohler und scharfkantiger Teile) über die Darstellung als tessellierte Oberfläche. Die Interferenz zwischen beliebigen Bauteilen wird im vorgestellten Ansatz durch das Volumen des Schnittbereichs der sich überlappenden Komponenten berechnet. Dafür wird auf eine externe Kollisionsbibliothek zugegriffen [Cohen et al., 1995]. Auch in [Yi et al., 2008] bei der Auslegung von Militärfahrzeugkonfigurationen wird zur Berechnung des Kollisionsvolumens von tesselierten Oberflächen eine externe Bibliothek [Corney und Lim, 2002] verwendet. Dabei ist zu erwähnen, dass die exakte Erkennung von Interferenzen sehr rechenintensiv sein kann. Der Vorteil dieses Vorgehens ist aber die quantitative Kollisionsauswertung, die eine Anwendung von gradientenbasierten Methoden erlaubt.

Hüllfläche und Hüllvolumen Die meisten bisher in der Literatur betrachteten dreidimensionalen Packingprobleme werden mit zumindest einigen quaderförmigen Approximationsgeometrien ausgeführt. In [Yi et al., 2008] beispielsweise werden viele der zu verteilenden Fahrzeugkomponenten als rechtwinklige Boxen repräsentiert. Auch in [Giorgio Fasano et al., 2014] und [Giorgio Fasano et al., 2015] werden Teile des Frachtguts in Form von rechtwinkligen Boxen in den Racks eines Raumfahrzeuges untergebracht. Für manche Anwendungen sind nicht mal Vereinfachungen notwendig, da tatsächlich quaderförmige Objekte in einem Bauraum untergebracht werden, wie beispielsweise bei der Bestimmung des Kofferraumvolumens von Fahrzeugen [Reichel, 2006]. Auch die sog. *container loading* Probleme sind klassische Packingproblemklassen, in denen quaderförmige Boxen optimal untergebracht werden müssen [Wu et al., 2017]. Eine weitere Alternative der Geometrievereinfachung ist die Verwendung von orbikularen Geometrien wie Kreisen oder Kugeln. Der Hauptvorteil der Verwendung von diesen besteht darin, dass die Kollisionsprüfung mit einfachen Abstandsberechnungen durchgeführt werden kann, wodurch das Problem grundlegenden Optimierungstechniken wie der mathematischen quadratischen Optimierung oder gradientenbasierten Methoden zugänglich wird. In der Literatur finden sich Ansätze mit Kreisen in [Jacquenot et al., 2009] und Ansätze mit Kugeln in [Zhu et al., 2012]. Auch [Jessee et al., 2020] und [Peddada et al., 2021a] verwenden für die Interferenzprüfung geometrische Vereinfachungen, wobei jede Systemkomponente als Hüllkreis repräsentiert wird und jedes verbindende Rohrelement als Linie. Für die Kollisionsfreiheit der Komponenten untereinander wird eine Kollisionsvermeidung von Kreisen eingesetzt. Zur Sicherstellung der Kollisionsfreiheit der Rohrelemente untereinander muss der minimale Abstand zwischen zwei Linien berechnet werden und einen vorgegebenen Mindestwert einhalten. Für die Prüfung eines Rohrelements zu einem Bauteil findet eine Kreis-Linien-Abstandsberechnung statt. Eine weitere beliebte Geometrievereinfachung ist die konvexe Hülle, welche eine deutlich genauere Approximation der Geometrie liefert als rechtwinklige oder orbikulare Geometrien,

gleichzeitig jedoch die positive Konvexitätseigenschaft erhält, welche vor allem für eine effiziente Kollisionsdetektion von Nutzen ist.

Pixelisierung und Voxelisierung Sollen die Nichtkonvexitäten der Geometrie trotz deren Vereinfachung erhalten bleiben, können auch die Pixelisierung oder Voxelisierung eingesetzt werden. Bei dem Packing von dreidimensionalen Kofferraumkomponenten in [Tiwari et al., 2008] und [Tiwari et al., 2010] werden Freiformgeometrien zunächst voxelisiert, um eine einfachere Kollisionsberechnung zu erhalten. Auch für die Konfiguration von Motorraumkomponenten wird in [Ravindranath, 2011], [Dong et al., 2011] und [Tiwari et al., 2014] dieselbe Strategie verfolgt. In [Ravindranath, 2011] wird der gesamte Motorraum als Bauraum global voxelisiert und die Kollisionserkennung prüft, ob Voxel unterschiedlicher Bauteile denselben Bauraumvoxel belegen. Die Genauigkeit der Interferenzdetektion ist dabei abhängig von der Voxelgröße. Ist die Voxelisierung fein genug, können dennoch sehr hohe Genauigkeiten erzielt werden, allerdings auf Kosten der Rechenzeit. Der Vorteil dieser Variante liegt in der Einfachheit der Kollisionsvermeidung, da lediglich die Pixel- oder Voxelbelegung ausgewertet werden müssen.

Verbund aus Basisobjekten [G. Fasano, 2015] befasst sich in seiner Veröffentlichung mit dem zweidimensionalen orthogonalen Packing von sog. *tetrisartigen* Elementen unter Berücksichtigung von rotatorischen Freiheitsgraden und zusätzlichen Nebenbedingungen in einem konvexen Gebiet. *Tetrisartig* beschreibt dabei Geometrien, welche durch Rechteckverbunde angenähert sind. Der vorgestellte Ansatz wird auf zwei Problemstellungen angewandt, zum einen die Maximierung der Anzahl der in das Gebiet hinein passenden rechtwinkligen Einzelelemente und zum anderen die optimale äußere Annäherung des polygonalen Gebietes über den Rechteckverbund. Nach demselben Vorbild werden auch in den Ansätzen von [Giorgio Fasano, 2013], [Giorgio Fasano et al., 2014] und [Giorgio Fasano et al., 2015] sowie [Gliozzi et al., 2015] und [Gliozzi et al., 2016] tetrisähnliche Geometrieannäherungen eingesetzt.

Bounding Volume Hierarchy Wenn mehrstufige Kollisionsüberprüfungen in Frage kommen, kann die numerische Komplexität weiter reduziert werden, wenn sogenannte *Bounding Volume Hierarchies* zur Diskretisierung der Komponentengeometrie verwendet werden. Diese Hüllvolumenhierarchien sind rekursive Baumdatenstrukturen, deren Blattknoten aus primitiven Hüllvolumina bestehen. Diese Knoten werden in Paketen gruppiert und in größere Hüllvolumina eingeschlossen, wodurch neue übergeordnete Knoten entstehen. Dies geschieht in rekursiver Weise und führt schließlich zu einer Baumstruktur. Der Hauptvorteil dieser Repräsentation besteht darin, dass Volumina nicht auf Kollisionen geprüft werden müssen, wenn ihr übergeordnetes Volumen nicht durchschnitten wird [Ericson, 2004]. Prinzipiell ist der Typ der Hüllvolumina dabei unerheblich, [Zhu et al., 2012] beispielsweise verwenden Kugelbäume als Geometrieapproximation für das effiziente 3D-Packing von unregelmäßig geformten Objekten.

3.3.2 Geometrierepräsentation als Optimierungsproblem

Bei der Lösung von Packingproblemen oder in anderen Worten der Packingoptimierung bestimmt die Güte der Geometrieapproximationen über die Güte des Packingverfahrens selbst. Geeignete Geometriemodelle und -repräsentationen aufzufinden, ist demnach ein zentraler Bestandteil des Lösungsprozesses, den es ebenso optimal zu lösen gilt. In diesem Sinne kann auch das Auffinden von ideal problemangepassten Geometrieapproximationen als Optimierungsproblem per se verstanden werden.

Wie bereits bei den Geometrievereinfachungen angeschnitten geben [G. Fasano, 2015] einige Einblicke in eine mögliche Vorgehensweise, tetrisartige Approximation unregelmäßiger Geometrien vorzunehmen. Die Autoren stellen einen auf einer gemischt-ganzzahligen linearen Optimierung basierenden Ansatz vor und beschränken sich auf den zweidimensionalen Fall konvexer Polygone. Das Problem wird darin so formuliert, dass ein beliebiges konvexes Polygon mit einem tetrisähnlichen Element mit minimaler Oberfläche bedeckt werden soll, das aus einer vorgegebenen Anzahl an Rechtecken besteht. Es liegt auf der Hand, dass eine bessere Annäherung an das Polygon möglich ist, je größer die Anzahl der Rechtecke ist. Nun könnte in der oben formulierten allgemeinen Problemstellung impliziert sein, dass die Abmessungen der einzelnen Rechtecke innerhalb bestimmter Bereiche mit Kontinuität variieren können [G. Fasano, 2015]. Die vorgestellte Methode beruht jedoch stattdessen auf einem vereinfachten Ansatz, in dem die Auswahl der Rechtecke auf eine endliche Anzahl von Möglichkeiten beschränkt ist, die sich aus einer a priori durchgeführten Diskretisierung ergeben.

Eine weitere veröffentlichte Vorgehensweise für die Modellierung einer Geometrievereinfachung als Optimierungsproblem kommt aus dem Bereich der rechnergestützten Bildverarbeitung. In [Demiröz et al., 2019] beschäftigen sich die Autoren mit dem sog. „Rectangle Blanket Problem“. Dieses beinhaltet die Ermittlung eines optimalen achsenorientierten Rechteckverbundes zur Vereinfachung einer zweidimensionalen Geometrie, wobei die sich nicht überlappenden Flächen zwischen Verbund und Geometrie minimiert werden sollen. Für die Berechnung wird die anzunähernde Geometrie zunächst pixelisiert und eine gemischt-ganzzahlige lineare Optimierungsmethode vorgestellt, in der die Pixel möglichst genau von dem zu optimierenden Rechteckverbund überdeckt sein sollen. Die gewünschte Anzahl der Rechtecke muss dabei vorgegeben werden. Der aus der Optimierung hervorgehende Verbund wird als eine sog. „Rechteckdecke“ bezeichnet und muss weder vollständig innerhalb der Geometrie enthalten sein, noch muss er die Geometrie vollständig bedecken.

3.3.3 Dynamische Geometriedimensionierung

Die Forderung nach einer ganzheitlichen Methodik für allgemeine Packingprozesse hat zur Folge, dass Systemkomponenten nicht ausschließlich positioniert und orientiert werden müssen, sondern auch eine Modifikation der Komponentenmaße ermöglicht werden muss. Im einfachsten Fall werden Freiheitsgrade für die Skalierung von Komponentengeometrien eingeführt, die jedoch auch in der Interferenzprüfung berücksichtigt werden müssen. In der Literatur existieren darüber hinaus auch komplexere Ansätze.

Die Dimensionierung über Skalierungsvariablen spielt bei der Erzeugung von Grundrissen im Bereich der Architektur und des Bauwesens eine zentrale Rolle, in denen Raumgrößen durch eine variable Breite und Länge optimiert werden können. In [Rodrigues et al., 2013a] beispielsweise wird jeder Raum durch ein Rechteck repräsentiert, dessen vertikale und horizontale Maße Teil der Optimierung sind. Die minimalen und maximalen Maße können dabei von dem/der Anwender/-in vor der Optimierung eingegrenzt werden.

In [Yin et al., 2004] z.B. wird ein erweiterter Mustersuchalgorithmus für die Platzierung von formbaren Komponenten in einem Produktlayout vorgestellt. Dabei werden formbare Octrees verwendet, um den Grad der Überlappung zwischen Bauteilpaaren zu bewerten. Die präsentierte Methode wird für den Entwurf eines Layouts eines Automatikgetriebequerschnitts angewendet. Die als *erweiterte Mustersuch-Layouttechnik* bezeichnete Methode bietet nach

Einschätzung der Autoren in Kombination mit der formbaren Octree-Darstellung ein effektives Werkzeug für das automatische Getriebe-Layout und andere Entwürfe, bei denen die Komponenten dynamisch dimensioniert werden müssen.

Im Beitrag von [Tiwari et al., 2014] wird im Kontext eines Motorraumpackings ein schneller und effizienter *Form-Morphing*-Algorithmus vorgestellt, der speziell für Packing- und Layout-Anwendungen entwickelt wurde. Dieser Algorithmus basiert auf einem modifizierten Masse-Feder-System, das zur Modellierung des morphbaren Objekts, im konkreten Beispiel des Wassertanks, verwendet wird und ahmt einen quasi-physikalischen Prozess nach, der dem Aufblasen und Entleeren eines mit Luft gefüllten Ballons ähnelt. Beginnend mit einer anfänglichen vielfältigen Geometrie, wird diese so gemorpht, dass ein gewünschtes Volumen erhalten wird, ohne dass die erhaltene Geometrie mit den sie umgebenden Objekten interferiert. Die Änderung der Geometrie wird durch die Simulation der Bewegung von Massepunkten an den Eckpunkten des Oberflächennetzes erreicht.

3.3.4 Geometrierepräsentation in einem Framework

Neben der Suche nach der besten geometrischen Repräsentation für eine gegebene packingspezifische Problemstellung bleibt die große Frage nach einer allgemeinen Repräsentationsform für generische Anwendungsfälle, die eine konsistente Kollisionsprüfung und die Einbettung in den Gesamtentwurf ermöglichen. Auch in der Literatur gibt es in diesem Kontext bereits Ansätze wie die Geometrierepräsentation bzw. deren Vereinfachung in einem Framework für die Lösung von allgemeinen Packagingproblemen umgesetzt werden könnte.

[Yakovlev und Kartashov, 2018] setzen sich mit eben dieser Thematik auseinander und schlagen eine analytische Beschreibung von Kollisionsbedingungen und Inklusionsbedingungen über die sogenannten *Phi-Funktionen* aus [Yurij Stoyan et al., 2004], [Scheithauer et al., 2005], [Chernov et al., 2010], und [Yuri Stoyan und Tatiana Romanova, 2013] vor. Zur Repräsentation von Geometrien werden in [Yuri Stoyan und Tatiana Romanova, 2013] die sogenannten *Phi-Objekte* vorgestellt, welche aus zweidimensionalen Basisgeometrien wie Kreisen, Ellipsen, Rechtecken und Polygonen und dreidimensionalen Basisgeometrien wie Kugeln, Kegeln, Zylindern, Quadern und Polyedern bestehen. Um der Repräsentation von komplexen Geometrien gerecht zu werden, werden zusätzlich komplexe Objekte eingeführt, die mithilfe von mengentheoretischen Operationen (Vereinigung, Schnittmenge, Additionen) aus Basisgeometrien aufgebaut werden können. [Yakovlev und Kartashov, 2018] stellen außerdem in ihrer Veröffentlichung fest, dass das Konzept der *Phi-Funktionen* für Objekte mit festen metrischen Parametern entwickelt wurde und daher eine Verallgemeinerung für den Fall erfordert, dass die geometrischen Maße der Objekte variabel sind.

[Yakovlev et al., 2018] stellen in ihrer Arbeit einen objektorientierten Ansatz für eine Repräsentationsstruktur von räumlichen Objekten vor. Sie entwickelten eine Typologie von einfachen und komplexen Objekten, die auf der Dimension des Raums und der Gestalt des Objekts basiert. In dieser ist eine allgemeine Klasse für ein Geometrieobjekt enthalten, welche gemeinsame Operationen für alle Objekte implementiert, wie z. B. das Abrufen von Informationen aus einer Datei, das Speichern einer Datei oder das Zeichnen auf dem Bildschirm. Weiterhin wird eine Gruppierung in drei Gruppen vorgenommen, die zweidimensionalen, dreidimensionalen und mehrdimensionalen Geometrien. Jede dieser Klassen enthält Felder und virtuelle Methoden, die eine affine Transformation der Bewegung im entsprechenden Raum ermöglichen. Auf der nächsten Ebene folgen die Unterklassen, die die Basisgeometrien des entsprechenden Raums

implementieren. Jede der Basisgeometrien ist dabei durch die Gleichung ihrer Begrenzung gegeben und enthält Informationen über ihre metrischen Eigenschaften. Auch hier findet eine Unterscheidung in einfache und komplexe Geometrien statt, eine komplexe Geometrie enthält für die Gestaltinformation alle Geometrien, aus denen sie zusammengesetzt ist. Ziel dieser Typologie ist die Ermöglichung, Objekte beliebiger Komplexität zu erstellen und zu verarbeiten.

3.4 Optimierungsverfahren für Packingprobleme

Wird eine allgemeine Packingaufgabe als globales Optimierungsproblem verstanden, besteht dieses darin, das globale Minimum oder Maximum einer Packingzielfunktion in einem n-dimensionalen Entwurfsraum zu finden. Die Zielfunktion und weitere Nebenbedingungen können dabei potenziell nicht differenzierbar, multimodal, selbst in einem Punkt teuer auszuwerten und sogar als Blackbox-Funktion gegeben sein. Nach [Sergeyev et al., 2018] werden traditionelle lokale Optimierungsmethoden diesen Ansprüchen nicht gerecht und sollten daher nicht verwendet werden. Unter den bestehenden gradientenfreien globalen Optimierungsverfahren bleiben drei Klassen von Algorithmen, die heuristischen Methoden, die stochastischen metaheuristischen Methoden und die deterministischen mathematischen Optimierungsverfahren. Heuristische Methoden finden sich in der Literatur, weil sie durch eine sehr hohe Problemanpassung besonders effizient für die vorgesehenen Problemstellungen sind. Genau diese Problemabhängigkeit ist allerdings auch der Grund für ihre fehlende Eignung für ein allgemeines Packingframework. Metaheuristiken dagegen werden aufgrund ihrer Einfachheit und ihrer attraktiven, von der Natur inspirierten Interpretationen von einer breiten Gemeinschaft von Ingenieuren/-innen und Praktikern/-innen zur Lösung realer Probleme eingesetzt [Sergeyev et al., 2018]. Ein besonderer Vorteil von Metaheuristiken ist, dass sie in Form von Metastrategien problemunabhängigen repräsentiert werden können und trotzdem Heuristiken für spezielle Anwendungsfälle ermöglichen. Die deterministischen mathematischen Verfahren wiederum werden aufgrund ihrer interessanten theoretischen Konvergenzeigenschaften vor allem in der Wissenschaft aktiv untersucht [Sergeyev et al., 2018]. Trotz der unterschiedlichen Natur aller drei Ansätze finden bei Packingproblemen in der Literatur häufig alle Varianten Anwendung. Im Folgenden soll hierzu ein Überblick gegeben werden.

3.4.1 Problemklassen

In der Literatur hat sich eine Klassifikation für mathematische Optimierungsprobleme etabliert, welche eine Einteilung in Abhängigkeit des Raumes der Entscheidungsvariablen, der Dimensionalität des Entscheidungsraums und der Nebenbedingungen und dem Zielfunktionstyp sowie der zulässigen Mengen vornimmt [Groell, 2018]. Der Raum der Entscheidungsvariablen definiert dabei, ob die eingehenden Variablen diskret oder stetig bzw. kontinuierlich sind oder ob es sich um gemischte Probleme handelt. Die Dimensionalität des Entscheidungsraums und der Nebenbedingungen ist eng an den Charakter der Variablen gekoppelt und wird auf der Basis der zulässigen Mengen in kombinatorische bzw. diskrete und finite Probleme aufgeteilt. Die kombinatorischen bzw. diskreten Probleme zeichnen sich dadurch aus, dass deren zulässige Menge mit nur endlich vielen Elementen gegeben ist, während finite Probleme durch die Kontinuität der Variablen theoretisch unendlich viele Lösungen zulassen. Hiernach könnte der Eindruck gewonnen werden, dass kombinatorische Probleme einfacher zu lösen sind als die finiten Varianten. Dem ist jedoch nicht so, da diskreten Mengen topologische Strukturen fehlen, die Ableitungskonzepte verfügbar machen, weshalb diskrete Probleme im Allgemeinen

nicht in Polynomialzeit gelöst werden können und als NP-schwer gelten [Groell, 2018]. Weiterhin wird nach dem mathematischen Grad in lineare und nichtlineare Varianten unterteilt sowie in konvexe und nichtkonvexe Kategorien. Diese Einteilung soll in der vorliegenden Arbeit beibehalten und um eine weitere Klasse für nicht rein mathematische Modelle erweitert werden, die sogenannte *Blackbox*-Klasse für Modelle, die in Form von Programmcode vorliegen. Die wichtigsten Auswirkungen der einzelnen Problemklassen auf Packingprobleme und deren Lösung sollen in den folgenden Abschnitten diskutiert werden.

Lineare oder Nichtlineare Formulierungen Obwohl Zielfunktionen und Restriktionen für Packingprobleme in der Regel von Natur aus nichtlinear und dadurch auch in nichtlinearer Form einfacher darstellbar sind (es bedarf schließlich keiner Änderung der physikalisch-mathematischen Modelle bei der Formulierung), werden sie in der Praxis häufig in ganzzahlig-lineare Repräsentationsformen überführt, was einen deutlichen Mehraufwand für die Modellierung bedeutet. Dies liegt in einfachen Worten daran, dass es nicht nur darum geht, die Modelle lediglich aufzustellen, sondern hauptsächlich darum, die Modelle auch zu lösen [Suhl und Mellouli, 2007]. Für lineare und gemischt-ganzzahlige Modelle existiert sowohl im kommerziellen wie auch im nichtkommerziellen Bereich leistungsfähige Standardsoftware, wodurch eine optimale Lösung meistens mit annehmbarem Rechenaufwand gefunden werden kann [Suhl und Mellouli, 2007]. Nicht umsonst sind lineare Optimierungsprobleme die am weitesten verbreitete Klasse von Optimierungsproblemen. Der enorme Erfolg lässt sich auf die Faktoren zurückführen, dass die erforderlichen Daten einfach sind, da sie ausschließlich aus Matrizen und Vektoren bestehen, dass die Konvexität garantiert ist, da das Problem konstruktionsbedingt konvex ist, dass lineare Funktionen trivial differenzierbar sind und die sogenannten Dualitätssätze (für weiterführende Informationen sei der interessierte Leser auf [Griva et al., 2008] verwiesen) eingesetzt werden können, wodurch besonders effiziente Algorithmen zur Lösung existieren [MOSEK ApS, 2019]. Optimierungssoftware für nichtlineare Probleme dagegen ist weitaus weniger vertreten, zudem ist die Lösung von nichtlinearen Problemen im Allgemeinen deutlich komplexer und zeitaufwändiger. Dadurch lohnt sich durchaus ein höherer Aufwand für die Überführung der ursprünglich nichtlinearen Modelle in die lineare Form. Selbst wenn die linearen Modelle dann nur eine Annäherung an das eigentliche Problem darstellen, können die Vorteile der linearen Optimierung die Nachteile des Genauigkeitsverlustes überwiegen [MOSEK ApS, 2019].

Konvexe oder Nichtkonvexe Probleme Da in allgemeinen Packingproblemen sowohl konvexe als auch nichtkonvexe Probleme vorhanden sein können, soll hier explizit auf deren Unterschied eingegangen werden. Diese Unterscheidung spielt eine wichtige Rolle, da die Konvexität oder Nichtkonvexität substanziell über die Komplexität eines Optimierungsproblems entscheidet. Das liegt daran, dass konvexe Probleme einige starke Eigenschaften aufweisen, die das Auffinden von globalen Optima erleichtern. Besonders ist dabei, dass jedes lokale Optimum des Problems auch immer ein globales Optimum darstellt. Bei nichtkonvexen Problemstellungen mag ein lokales Optimum zwar schnell gefunden sein, der Nachweis der globalen Optimalität wird allerdings extrem schwierig. Daher müssen nichtkonvexe Probleme zur Lösung häufig in konvexe Darstellungen umgeformt (konvexifiziert) werden. Glücklicherweise lässt sich ein Großteil original nichtkonvexer Probleme dank eines großen Repertoires an Techniken konvexifizieren [Groell, 2018]. In der Vergangenheit wurden Optimierungsprobleme häufig in lineare und nichtlineare Komplexitätsklassen unterteilt. Mit fortschreitender Forschung hat sich dies allerdings bereits in den 90ern verändert. Dies spiegelt sich auch in folgendem Zitat von [Rockafellar, 1993] wider: „...in fact, the great watershed in optimization isn't between linearity and nonlinearity, but convexity and nonconvexity.“. Für die Modellierung von Optimierungsmodellen

sei darauf hingewiesen, dass konvexe Formulierungen dennoch so gut wie immer nichtkonvexen Formulierungen vorzuziehen sind. Der Vollständigkeit halber sei außerdem gesagt, dass dieselben Aussagen für konkave Probleme gelten, es wird also keine Unterscheidung zwischen konvex und konkav gemacht, viel mehr wird zwischen konvex bzw. konkav und keiner der beiden Eigenschaften unterschieden. Überprüft werden kann die Konvexität unter Annahme einer zweimal stetig differenzierbaren Funktion mithilfe ihrer Hessematrix. Positive Semidefinitheit der Matrix lässt auf Konvexität schließen, positive Definitheit lässt sogar auf strikte Konvexität schließen. Genauso lässt negative Semidefinitheit der Matrix auf Konkavität schließen und negative Definitheit sogar auf strikte Konkavität.

Diskrete und kontinuierliche Modelle Bei allgemeinen Packingproblemen sind sowohl diskrete als auch kontinuierliche Modelle beteiligt, welche in der Regel in Mischformen vorliegen. Die meisten physikalischen Zusammenhänge werden über kontinuierliche Modelle repräsentiert. Dort stellt der zulässige Bereich der potenziellen Lösungen ein Kontinuum dar. Im Gegensatz dazu stehen die diskreten Modelle, welche ausschließlich ganze Zahlen anstelle von Dezimalzahlen verwenden. Diese werden vor allem bei topologischen Fragestellungen eingesetzt, da im Allgemeinen davon ausgegangen werden kann, dass auf Basis kontinuierlicher Modelle keine topologischen Veränderungen vorgenommen werden können. Die zwingende Existenz von diskreten Modellen wird besonders deutlich, wenn man sich vor Augen führt, dass Packingaufgaben vor allem bei der Optimierung von Konfigurationen mit Entscheidungen zu tun haben. Dabei können Komponenten ausgewählt werden oder nicht ausgewählt werden. Auch viele der zum Einsatz kommenden Interferenzmodelle prüfen lediglich, ob eine Kollision existiert oder nicht und geben keine weiteren kontinuierlichen Informationen darüber aus. Die diskreten Modelle können hierbei noch weiter unterteilt werden, in ganzzahlige (beispielsweise bei der Auswahl einer bestimmtem Topologie aus einem Set an möglichen Varianten) und kombinatorische Modelle (bei der Repräsentation von *booleschen* Entscheidungen). Dabei ist die Unterscheidung in kontinuierliche und diskrete Problemstellungen keineswegs künstlich, da sowohl Strukturresultate, als auch Methodik und Algorithmen für ihre Lösung grundsätzlich verschieden sind [Weismantel, 2003]. Für allgemeine Packingprobleme werden also sowohl diskrete Optimierungsverfahren als auch kontinuierliche Optimierungsverfahren benötigt.

Mathematische Funktionen oder Blackbox-Funktionen Viele packingspezifische Fragestellungen können durch eine smarte Modellierung in Form von mathematischen Modellen repräsentiert werden. Dennoch können Problemstellungen auftreten, die nicht rein mathematisch darstellbar sind, beispielsweise wenn Ergebnisse nur über iterative Methoden ausgewertet oder wenn bestimmte Variablen nur über Simulationen berechnet werden können. Solche Fälle werden in der Literatur häufig als Blackbox-Funktionen bezeichnet. Aber auch ganz pragmatische Motive können zu dem Vorhandensein von Blackbox-Funktionen führen, beispielsweise wenn bereits implementierte Verfahren wiederverwendet werden sollen, die für einen gegebenen Input einen Output berechnen, ohne dass die Berechnungsmethodik bekannt ist. In diesem Kontext ist besonders wichtig, dass deterministische mathematische Optimierungsverfahren zwingend eine mathematische Formulierung erfordern. Sollen Blackbox-Funktionen eingesetzt werden, können zur Lösung des Gesamtproblems ausschließlich metaheuristische Verfahren verwendet werden (oder heuristische, welche sich allerdings aufgrund der Problemabhängigkeit wenig für ein allgemeines Packingframework eignen).

3.4.2 Deterministische mathematische Optimierung

Das Lösen eines Optimierungsproblems ist ein wesentlicher Bestandteil bei der Automatisierung von Packingaufgaben. Die Mathematik liefert hierfür die Theorie und auch die Algorithmen, auf die der/die Anwender/-in dank der Verfügbarkeit sowohl kommerzieller als auch freier verfügbarer Software zugreifen kann [Groell, 2018]. Werden also Aufgaben aus dem Packing als mathematisches Optimierungsproblem abgebildet, fällt dies in den Bereich der deterministisch mathematischen Optimierung. Bei der Modellierung dieser Art von Optimierungsproblemen kommt es besonders darauf an, die packingspezifischen Fragestellungen so abzubilden, dass sie in eine mathematische Problemklasse fallen, für die möglichst starke theoretische Aussagen existieren und für die numerisch stabile Algorithmen zur Verfügung stehen [Groell, 2018]. Dem/der Anwender/-in kommt dabei ein Trend der letzten Jahre zugute, der durch die Schaffung mächtiger Werkzeuge zur Problemformulierung geprägt ist [Groell, 2018]. Die entstandenen algebraischen Modellierungssprachen gestatten, unterschiedliche Löser einzubinden und unterstützen zudem teilweise sogar die effiziente Ableitungsberechnung oder numerische Integration. Die Voraussetzung für die Anwendung von mathematischen Optimierungsverfahren ist allerdings ein gewisses mathematisches Grundverständnis im Bereich der Optimierung. Typische Methoden zur Problemlösung sind die *Simplex*-Methoden, darunter der *Primal-*, *Dual-* und *Network-Simplex* Ansatz ([Stein, 2021], [Stein, 2018], [Schwenkert und Stry, 2015], [Hochstättler, 2010], [Hamacher und Klamroth, 2006]), das sog. *Sifting* als Erweiterung zu den *Simplex*-Methoden, *Interior Point-* oder *Newton Barrier*-Methoden, mit oder ohne *Crossover*-Techniken ([Stein, 2021], [Stein, 2018], [Vanderbei, 2013], [Ge et al., 2021]) und nicht zuletzt die *Branch and Bound*-Verfahren ([Stein, 2021], [Stein, 2018]). Für einen Überblick sollen an dieser Stelle einige Arbeiten aus der Literatur vorgestellt werden, die im letzten Jahrzehnt entstanden sind.

In der Veröffentlichung von [Paquay et al., 2014] wird eine gemischt-ganzzahlige Formulierung für ein Problem aus der Luftfrachtanwendung vorgestellt, in dem Frachtgut in sog. Ladeeinheiten von Flugzeugen untergebracht werden soll. Die entwickelte Formulierung berücksichtigt sowohl dem Transport innewohnende Beschränkungen wie die Stabilität und Zerbrechlichkeit der Ladung als auch systemrelevanten Bedingungen für die Gewichtsverteilung.

Die Arbeit von [G. Fasano, 2015] dürfte dem Leser bereits in Abschnitt 3.3.1 begegnet sein. Dort präsentieren die Autoren ein Problem, bei dem tetrisartige Elemente innerhalb von konvexen Gebieten berechnet werden. Das Gesamtproblem wird von ihnen sowohl als gemischt-ganzzahliges lineares als auch als gemischt-ganzzahliges nichtlineares Problem formuliert und unter Einsatz der mathematischen Optimierungsverfahren gelöst. Die vorgestellte Arbeit soll dabei den Startschuss zu weiteren zukünftigen Untersuchungen bilden, die sich auf die Anwendung von diskreten mathematischen Optimierungsstrategien konzentrieren.

In den Forschungsarbeiten von [Giorgio Fasano, 2013] beschäftigen sich die Autoren im Bereich der Raumfahrttechnik mit der Frachtunterbringung in Raumfahrzeugen und deren Modulen. Ausgehend von der Idee, dass sich Frachtgeometrien als tetrisartige Elemente, d. h. als Gruppen von zueinander orthogonalen Parallelepipeden, darstellen lassen, wird die orthogonale Positionierung von tetrisartigen Elementen in einem konvexen Gebiet untersucht. Zur Lösung werden gemischt-ganzzahlige lineare und nichtlineare Optimierungsmethoden verwendet. Ziel des Ansatzes ist die Ausnutzung von vorteilhaften linearen Strukturen in den entsprechenden mathematischen Modellen. In einem weiteren Testfall wird die Positionierung von Polygonen mit kontinuierlichen Rotationen innerhalb eines konvexen Gebietes betrachtet.

Basierend auf den Erkenntnissen aus [Giorgio Fasano, 2013] wird in [Giorgio Fasano et al., 2014] und [Giorgio Fasano et al., 2015] ein Hilfsprogramm für die Frachtunterbringung in Raumfahrzeugen unter dem Namen *CAST (Cargo Accommodation Support Tool)* vorgestellt. Dieses wurde entwickelt, um die analytische Frachtunterbringung in einem *Automated Transfer Vehicle* Transportsystem für alle seinen Missionen zu unterstützen und basiert auf einer Bibliothek aus problemspezifischen gemischt-ganzzahligen linearen Optimierungsmodellen und heuristischen Algorithmen.

[Gliozzi et al., 2015] und [Gliozzi et al., 2016] beschäftigen sich ebenfalls vor dem Hintergrund der Beladung von Containern und Fahrzeugen sowie der Konfiguration von Satelliten und Raumfahrzeugen mit einer Erweiterung des klassischen *Container*-Problems, in der nicht ausschließlich rechtwinklige Basisobjekte wie Boxen, sondern tetrisähnliche Geometrieapproximationen von komplexen Geometrien in vorgegebene Behälter gepackt werden. Das zugrundeliegende mathematische Modell wird dort als gemischt-ganzzahliges lineares Optimierungsmodell formuliert und kombiniert mit einer Heuristik unter Einsatz einer passenden Lösungssoftware berechnet. Aus Sicht der Autoren ebnen die präsentierten Ergebnisse den Weg für eine vielversprechende weiterführende Forschung.

In [Grebennik et al., 2018] wird die Optimierung der Gewichtsverteilung für eine gegebene Menge von dreidimensionalen Objekten in einem Container betrachtet, der durch horizontale Regale in Untercontainer unterteilt ist. Für die analytische Beschreibung der Kollisionsfreiheit und der Inklusionsregeln werden die sog. *phi-Funktionen* (siehe Abschnitt 3.3.4) verwendet. Es werden Konfigurationen definiert, die die kombinatorische Struktur des Problems beschreiben. Auf deren Grundlage wird ein mathematisches Modell konstruiert, das nicht nur die Positionsbedingungen und die mechanischen Eigenschaften des Systems berücksichtigt, sondern auch die kombinatorischen Eigenschaften, welche mit der Einteilung der in den Subcontainern unterzubringenden Objekte zusammenhängen.

Der Beitrag von [Y. Stoyan et al., 2016] untersucht das Problem der optimalen Anordnung von dreidimensionalen Objekten (Vollkugeln, geraden Kreiszylindern, Kugelzylindern, geraden regelmäßigen Prismen, Quadern und Tori) in einem Behälter (zylindrisch, parabolisch oder kegelstumpfförmig) mit kreisförmigen Racks. Die Problemformulierung berücksichtigt einen vorgegebenen minimalen und maximalen Abstand zwischen den Objekten sowie die mechanischen Nebenbedingungen des Gesamtsystems hinsichtlich des Gleichgewichts, der Trägheitsmomente und der Stabilität. Zur Lösung wird unter Verwendung der *phi-Funktionen* ein mathematisches kontinuierliches nichtlineares Optimierungsproblem formuliert.

Auch [T. Romanova et al., 2018] stellen eine auf *phi-Funktionen* basierende Methode vor, um eine gegebene Sammlung von beliebigen, im Allgemeinen konkaven Polyedern in einen Quader mit minimalem Volumen zu packen, wobei kontinuierliche Rotationen und Translationen erlaubt sind. Unter Berücksichtigung der Kollisionsfreiheit und vorgegebener Abstände wird für die Repräsentation des Gesamtproblems ein exaktes mathematisches nichtlineares Modell abgeleitet. Die Autoren entwickeln außerdem ein Verfahren, welches das Gesamtproblem auf eine Folge von Teilproblemen mit wesentlich geringerer Dimension und einer geringeren Anzahl von nichtlinearen Ungleichungen reduziert.

[Cherri et al., 2019] nutzen die deterministische mathematische Optimierung für zweidimensionale Packingprobleme im Bereich der Produktion von Zuschnittware. Die Autoren präsentieren eine gemischt-ganzzahlige Problemformulierung für die Positionierung der Zuschnitte auf der Rohware unter Berücksichtigung der Überlappungsfreiheit, wobei die translatorischen und rotatorischen Freiheitsgrade sowie die Auswahl der Zuschnitte über binäre und diskrete ganzzahlige Variablen stattfindet. Für die Berücksichtigung von kontinuierlichen rotatorischen

Freiheitsgraden stellen [Cherri et al., 2018] einen Ansatz mit einer gemischt-ganzzahlige Formulierung mit quadratischen Nebenbedingungen vor.

3.4.3 Metaheuristische Optimierungsverfahren

Eine Metaheuristik ist ein übergeordnetes, problemunabhängiges algorithmisches Framework, das eine Reihe von Leitlinien oder Strategien für die Entwicklung heuristischer Optimierungsalgorithmen bereitstellt [Sörensen und Glover, 2013]. Metaheuristische Algorithmen sind in den meisten Fällen von stochastischer Natur, was sie von exakten Methoden unterscheidet, die einen Beweis erbringen, dass die optimale Lösung in einer endlichen, wenn auch oft untragbar großen, Zeitspanne gefunden wird. Daher werden sie meistens dann verwendet, wenn eine Lösung in einer akzeptablen Rechenzeit gefunden werden muss, die lediglich „gut genug“ ist. Sie erfreuen sich im technischen Umfeld großer Beliebtheit, was auch daran liegt, dass sie nicht der kombinatorischen Explosion unterliegen, oder in anderen Worten, dem Phänomen, dass die für die optimale Lösung eines NP-schweren Problems benötigte Rechenzeit exponentiell mit der Problemgröße ansteigt. Viele metaheuristische Ansätze sind von der Natur inspiriert, beispielsweise die populationsbasierten Ansätze wie die *genetische* oder *evolutionäre* Optimierung, die *Partikel-Schwarm*-Optimierung und Derivate davon, der *Ameisenkolonie*- oder *Bienen*-Algorithmus, der *Glühwürmchen*-Algorihtmus, der *Fledermaus*-Algorithmus und die vom Kuckuck inspirierte *cuckoo*-Suche oder die *Harmony Search*-Methode, aber auch lokale Suchverfahren wie die *Simulated-Annealing*-Methode oder andere Mustersuchalgorithmen sowie viele weitere lassen sich den Metaheuristiken zuordnen. Ein weiterer Vorteil von Metaheuristiken ist, wie der Name bereits impliziert, dass sie bei der konkreten Implementierung mit Heuristiken befüllt werden und dadurch an die spezifischen Fragestellungen angepasst werden können. Aufgrund ihrer einfachen Anwendbarkeit und Anpassbarkeit sind sie daher bei einer Vielzahl von packingverwandten Fragestellungen in der Literatur zu finden.

Wie etwa in der Studie von [Ravindranath, 2011], in welcher der Autor einen agentenbasierten Ansatz zur Lösung des Packings von Motorraumkomponenten anwendet. Im formulierten System werden drei Zielkriterien berücksichtigt, darunter der Schwerpunkt, die Überlebensfähigkeit des Fahrzeugs und die Wartungsfreundlichkeit. Modellierte Nebenbedingungen umfassen die Überschneidungsfreiheit zwischen den Komponenten und von Komponenten mit dem Gehäuse sowie eine vorgegebene minimale Bodenfreiheit. Der agentenbasierte Ansatz wird konkret als nicht-deterministischer evolutionärer Mehrzielalgorithmus implementiert, welcher nicht-dominierte Lösungen identifizieren kann und dadurch die Konvergenz zu einer nicht-dominanten suboptimalen Lösungsmenge beschleunigt. Das entwickelte Modell wird auf einen Personenkraftwagen angewandt, kann aber nach Aussage des Autors auch für das Packing von größeren Fahrzeugen eingesetzt werden, wie für Geländewagen und Lastkraftwagen. Insgesamt demonstriert die vorgestellte Methodik die prinzipielle Anwendbarkeit eines agentenbasierten Ansatzes für Packingprobleme im Automobilbereich.

Auch [Grignon und Fadel, 2004], [Tiwari et al., 2008], [Dong et al., 2011], [Katragadda et al., 2012], [Dandurand et al., 2014], [Tiwari et al., 2014] und [Fadel und Wiecek, 2015] untersuchen Packingprobleme im Automobilsektor, die Autoren setzen in ihren Arbeiten auf genetische Methoden für die Optimierung. In demselben Umfeld setzen sich [Tiwari et al., 2010] mit der Eignung von genetischen Optimierungsalgorithmen für die Unterbringung von Freiformobjekten innerhalb eines beliebig geformten Behälters auseinander. Dafür stellen sie drei Untersuchungsreihen vor, in denen erstens rechtwinklige Boxen in einem rechtwinkligen Bauraum positioniert werden, zweitens rechtwinklige Boxen in einem geometrisch nichtkonvexen Bauraum untergebracht werden und letztlich Freiformobjekte mit voller rotatorischer

Freiheit in einem nichtkonvexen Bauraum positioniert werden. Für eine ausführliche Untersuchung wird jede Testreihe mit einer Vielzahl unterschiedlicher zufälliger Initialparameter für die genetische Optimierung durchgeführt.

[Fadel und Wiecek, 2015] argumentieren in ihrer Arbeit zur Packingoptimierung von Freiformobjekten im Ingenieurentwurf, dass der Komplexitätsgrad von Packingproblemen die Entwicklung exakter Optimierungsmethoden ausschließt. Den Autoren nach sind evolutionäre Algorithmen und insbesondere genetische Ansätze oder Simulated-Annealing-Verfahren die einzigen Ansätze, die diese Schwierigkeiten lösen können. In den Augen der Autoren haben sich genetischen Algorithmen als attraktives Optimierungsinstrument für eine breite Klasse von Einziel- und Mehrzielproblemen erwiesen, ihre Eignung für Packingprobleme rechtfertigen die Autoren mit der Notwendigkeit, neben funktionalen Zielformulierungen auch Pareto-optimale Lösungen zu erzeugen.

Genetische Algorithmen werden auch bei architektonischen Fragestellungen eingesetzt. Ziel der Studie von [S. Lee und K. S. Lee, 2020] ist die Entwicklung einer Methode, die bei der Planung von Wohnkomplexen im städtischen Kontext hilft, indem sie sich neben der bloßen Verringerung des Schattenwurfs zwischen Gebäuden auch auf die Tageslichtumgebung in Innenräumen konzentriert. Nach Empfehlung der Autoren kann die entwickelte simulationsbasierte genetische Optimierung in den ersten Planungsphasen eines integrierten nachhaltigen Entwurfs in einer Stadt mit hoher Bevölkerungsdichte eingesetzt werden.

In [Guo und Li, 2017] wird eine Methode zur automatischen Generierung eines dreidimensionalen Architekturlayouts bzw. -packings aus einem benutzerdefinierten Architekturkonzept vorgestellt. Der vorgeschlagene Ansatz verbindet ein Multi-Agenten-Topologie-Findungssystem und einen evolutionären Optimierungsprozess. Ersteres erzeugt topologisch zufriedenstellende Layouts für die weitere Optimierung, während letzteres sich auf die Optimierung der Layouts konzentriert, um vordefinierte architektonische Kriterien zu erreichen. Der Topologiefindungsprozess engt dabei den Suchraum bereits vorab ein und erhöht so die Leistung bei der anschließenden Optimierung. In den Augen der Autoren zeigen die Ergebnisse, dass der vorgestellte Ansatz für eine dreidimensionale mehrstöckige Auslegung von Gebäuden geeignet ist.

Für zweidimensionale Probleme mit rein geometrischen Nebenbedingungen stellen [Magalhaes-Mendes et al., 2017] im Kontext der Fertigung und genauer bei dem Zuschnitt von zweidimensionaler Rohware wie Blechen oder textilen Materialien einen Ansatz vor, welcher auf einem genetischen sog. parallelen *Biased Random-Key* Algorithmus mit mehreren Populationen basiert. Ziel ist die Auffindung einer effiziente und schnelle Layout-Lösung für eine maximale Ausnutzung der flächigen Rohware.

In der Arbeit von [Sridhar et al., 2017] wird für den Transport- und Logistikbereich ein genetischer Algorithmus zur Lösung des Packingproblems von dreidimensionalen Behältern beliebiger Größe in rechtwinkligen prismatischen Behältern vorgestellt, bei dem die meisten praktischen Einschränkungen der Logistikindustrie berücksichtigt werden. Ziel des vorgestellten Ansatzes ist, eine Anzahl von Kisten mit beliebiger Größe so in einem Behälter mit Standardabmessungen unterzubringen, dass die Volumennutzung und damit der Gewinn maximiert werden. Für ein praxistaugliches Packschema werden verschiedene praktische Restriktionen berücksichtigt wie die Ausrichtung der Kisten, Stapelprioritäten, die Stabilität des Behälters, Gewichtseinschränkungen und Überlappungseinschränkungen.

Auch bei der Fertigungsplanung können Layoutprobleme mithilfe metaheuristischer Methoden gelöst werden. [Jingfa Liu und Jun Liu, 2019] und [McKendall und Hakobyan, 2021] beschäftigen sich unabhängig voneinander mit der Optimierung von Fertigungslayouts. [Jingfa Liu

und Jun Liu, 2019] verwenden eine modifizierte Version der multikriteriellen Ameisenkolonie-Optimierung unter Berücksichtigung der Überlappungsfreiheit von unterschiedlichen Fertigungsbereichen. [McKendall und Hakobyan, 2021] stellen ein mathematisches Modell und einen genetischen Algorithmus für die Positionierung von Anlagen in einer Fabrikhalle vor, wobei die Minimierung der Gesamtstrecke des Materialflusses zwischen den Anlagen vorgesehen ist.

In der Studie von [Carta et al., 2020] wird ein Optimierungsansatz für die Raumaufteilung bei der Planung von Pflegeheimen vorgestellt und erörtert. Die Autoren stellen eine Methode zur Steigerung der Effizienz des Grundrisses mithilfe eines selbstorganisierenden genetischen Algorithmus vor, die den Energieverbrauch senkt, das Wohlbefinden der Bewohner verbessert und sich implizit auf die Energie- und Gesundheitskosten auswirkt. Um eine optimale räumliche Konfiguration zu finden, wird eine Reihe von Gestaltungskriterien ausgearbeitet, welche als Ziele in einem genetischen Algorithmus verwendet werden, um die beste Lösung zu bewerten. Innerhalb der Iterationen des genetischen Algorithmus wird zur Generierung der Flure eine Ameisenkolonie-Optimierung verwendet. Der selbstorganisierte Grundriss wird dann zur Durchführung einer abschließenden Simulation verwendet, um die Effizienz der automatisiert generierten Pläne zu messen.

Ein Beispiel für den Einsatz einer Partikel-Schwarm-Optimierung ist die Veröffentlichung von [Jacobson, 2017], in welcher eine Verallgemeinerung der selbstähnlichen Verschachtelung der sog. *Matrjoschka-Puppen* auf beliebig geformte Körper untersucht wird. Die Autoren befassen sich mit dem Problem, die größte Kopie eines Objekts zu finden, welche vollständig in das originale Objekt hineinpasst, wobei es außerdem möglich sein soll, das größere Exemplar in zwei Teile zu schneiden und das kleinere Objekt ohne Kollisionen zu entfernen. Die Autoren verwenden eine Partikel-Schwarm-Optimierung, da sie die Berücksichtigung verschiedener Freiheitsgrade erlaubt, darunter eine rotationsfreie Bewegung des inneren Objekts bei dessen Entfernen, die Entfernungsrichtung, die Ausrichtung der Schnittebene und die minimale Wandstärke. Ziel bei der Methode ist die Maximierung der Größe des inneren Objekts bei der validen Verschachtelung von sowohl gleichartigen wie auch unterschiedlichen Geometrien.

Auch Mustersuchstrategien können als Metaheuristiken eingesetzt werden, z.B. für die effiziente Optimierung von Bauteilanordnungen. [Yin und Cagan, 2000] führen hierfür einen Mustersuchalgorithmus für beliebige Bauteilgeometrien, vielfältige Entwurfsziele und weitere räumlichen Beschränkungen ein. Der präsentierte Algorithmus wird im Verlauf der Veröffentlichung erweitert, um ein gleichzeitiges Layout- und Routing-Problem zu lösen, was nach Ansicht der Autoren die Flexibilität des Algorithmus demonstriert. Als Ergänzung untersuchen [Yin und Cagan, 2004] unterschiedliche Heuristiken, die in Mustersuchmethoden zur Identifizierung vielversprechender Suchrichtungen eingesetzt werden können. Durch die Untersuchung verschiedener Suchmuster und ihrer Wirksamkeit auf die Lösung von Packingproblemen wird die Frage behandelt, ob komplexe Taktiken besser abschneiden können als eine einfache Koordinatenmustersuche. Spätere Untersuchungen umfassen die Modifikation der Reihenfolge, in der Muster während der Suche eingeführt werden [Aladahalli et al., 2007] und den Vorschlag einer Sensitivitätsmetrik zur Abschätzung der Auswirkung von Musterbewegungen auf die Zielfunktion zur Berechnung der optimalen Reihenfolge [Aladahalli et al., 2005].

Ein auf Simulated-Annealing basierender Ansatz wird in der Forschungsarbeit von [Szykman und Cagan, 1997] vorgestellt, um dreidimensionale Bauteilkonfigurationen zu optimieren. Im vorgestellten Ansatz berücksichtigte Nebenbedingungen sind das Erreichen einer hohen Packungsdichte, das Einpassen von Bauteilen in einen gegebenen Behälter und die Erfüllung von räumlichen Beschränkungen die sich aus den Abhängigkeiten der Bauteile voneinander ergeben. Zur Veranschaulichung des entwickelten Algorithmus wird er auf die Positionierung von Komponenten einer Akku-Bohrmaschine angewendet.

In [Zheng und Ren, 2020] wird ein ebenfalls auf der Simulated-Annealing Methode basierendes System zur Erstellung eines architektonischen Layouts entwickelt, das Architekten in einem frühen Stadium der Gebäudeplanung unterstützen soll. In der vorgestellten Arbeit werden zur Bewertung eines Entwurfsplans sechs Ziel- und Straffunktionen mit unterschiedlichen Gewichtungen parametrisiert, die als Leitfaden für die architektonische Gestaltung, insbesondere für die Planung von Wohngebieten, dienen. Die mithilfe der Funktionen entwickelte Bewertung wird in der anschließenden Simulated-Annealing-Lösungssuche herangezogen, um automatisch erstellte Entwurfspläne zu optimieren.

Die Autoren von [Allahyari und Azab, 2018] zeigen sogar, dass die Simulated-Annealing Methode für bestimmte Fälle potenziell besser geeignet sein kann als die mathematische Optimierung. In ihrer Arbeit formulieren die Autoren ein gemischt-ganzzahliges nichtlineares Modell für die Positionierung von unterschiedlich großen rechteckigen Anlagen innerhalb eines ebenen Werksgeländes mit einer vorgegebenen festen Fläche. Für die Anlagen werden Beschränkungen entwickelt, um mögliche Überschneidungen zu eliminieren. Außerdem berücksichtigt das Modell sowohl vertikale als auch horizontale Gänge sowie Blöcke und Vorzugsplätze, an denen keine Anlagen platziert werden dürfen. Die Zielfunktion repräsentiert die Gesamtkosten für den Materialumschlag und soll minimiert werden. Bei der Lösung stößt die vorhergesehene mathematische Optimierung allerdings auf ihre Grenzen, sodass ein neuer Ansatz mithilfe der Simulated-Annealing Methode entwickelt werden muss, welcher schließlich erfolgreich zur Lösung des Problems eingesetzt wird.

3.4.4 Heuristische Verfahren

Heuristische Verfahren bestehen üblicherweise aus einer Reihe problemangepasster Regeln, welche aus Erfahrungen abgeleitet werden oder einfach auf logischen Zusammenhängen des Problems basieren. Heuristiken sind in der Lage Lösungen in moderaten Rechenzeit zu liefern, weshalb sich trotz ihrer fehlenden Generizität ein Blick in die Literatur lohnt, da sie Einblicke in die Mechanismen von effizienten Packingverfahren geben. Überzeugende Lösungen können zur konkreten Umsetzung von Metaheuristiken verwendet werden oder um komplexe Packingprobleme durch Einsatz von hybriden Verfahren zu vereinfachen. Die meisten neueren Ansätze für Packingprobleme mit hoher Komplexität, die in der Literatur zu finden sind, basieren schon seit langem auf hybriden Techniken [Cagan et al., 2002].

Vor dem wissenschaftlichen Hintergrund der optimalen Positionierung von Objekten im Inneren eines Satellitengehäuses untersuchen [Teng et al., 2001] Layoutprobleme bei denen Objekte im Inneren eines konischen, sich spiralförmig bewegenden Gehäuses angeordnet werden sollen. Dabei sollen Kollisionsfreiheit, optimale Raumausnutzung und die durch die Rotation des Gehäuses verursachte Unwucht infolge eines dynamisches Ungleichgewichts berücksichtigt werden. Es wird eine iterative zweistufige Lösungsstrategie vorgeschlagen, die ein vorläufiges 2D-Optimierungsproblem und ein darauf aufbauendes 3D-Optimierungsproblem umfasst. Das vorläufige zweidimensionale Problem wird außerdem in eine initiale und eine optimale Layoutphase unterteilt. In der initialen Layoutphase wird mithilfe einer heuristischen Methode entschieden, auf welcher der beiden Satellitenplattformen sich die Objekte befinden, woraufhin eine zweidimensionale Positionierung mithilfe von gradientenbasierten Heuristiken erfolgt, die ein initiales Layout liefert. Die dreidimensionale Layout-Optimierung erfolgt durch den Austausch der Objekte zwischen den beiden Basen. Alle Schritte werden so lange durchgeführt, bis ein Gesamtoptimum erreicht ist.

Auch [Zhi-Guo und Hong-Fei, 2003] beschäftigen sich mit der Optimierung der Konfiguration von Kommunikationssatelliten in Hinblick auf das Trägheitsverhalten. Sie schlagen eine zweistufige heuristische Lösungsstrategie vor, in welcher der Entwurfsprozess in einen vorläufigen Vorentwurf und einen anschließenden Detailentwurf zerlegt wird. Im Vorentwurf werden nur die Trägheitsmomente bezüglich der x- und y-Achse betrachtet, sodass nur ein eindimensionales Problem gelöst werden muss: die Verteilung der Objekte entlang der z-Achse, was im Konkreten die Verteilung der Objekte auf vier Auflageflächen bedeutet. Hierfür wird eine Heuristik vorgeschlagen, welche als *zentripetale Ausgleichsmethode* bezeichnet wird. Erst im Anschluss erfolgt die dreidimensionale, detaillierte Platzierung der Komponenten auf den zuvor ausgewählten Flächen mithilfe eines metaheuristischen Optimierungsansatzes.

Für eine Reduktion der Gesamtkomplexität beinhaltet eine Vielzahl der in der Literatur vorgestellten Heuristiken die vorgeschaltete Berechnung einer Reihenfolge, in der Komponenten nacheinander positioniert werden. In dem aus den vorigen Abschnitten bereits bekannten Ansatz von [Yi et al., 2008] werden die Fahrzeugkomponenten nacheinander in einer vorab definierten Reihenfolge positioniert. Auch die bereits genannten Arbeiten von [Tiwari et al., 2008] und [Tiwari et al., 2010] setzen eine vorgeschaltete Optimierung der Einbaureihenfolge und der Komponentenausrichtung mithilfe eines genetischen Algorithmus ein. Die anschließende, auf der berechneten Reihenfolge basierende Positionierung bedient sich außerdem der sogenannten *Bottom-Link-Back-Fill*-Heuristik, bei der die Positionierung eines Objekts von der unteren linken hinteren Ecke des Containers ausgeht, welches so lange verschoben wird, bis eine geeignete Position gefunden ist, die sich weder mit dem Container noch mit den bereits platzierten Objekten überschneidet [Fadel und Wiecek, 2015].

Auch [Dong et al., 2011] und [Tiwari et al., 2014] müssen aufgrund der hohen Komplexität ihrer Problemmodelle heuristische Vereinfachungen einführen. Ihre Bemühungen, ein All-in-One-Problem zu lösen, stellen sich als erfolglos heraus, da der genetische Optimierungsalgorithmus nicht in der Lage ist, die Komponenten gleichzeitig zu platzieren und die variable Form des Morphing-Objekts so zu verändern, dass ausschließlich der verfügbare Platz belegt wird [Fadel und Wiecek, 2015]. Daher setzen die Autoren auf eine Aufspaltung des Gesamtproblems in ein mehrstufiges Verfahren, wobei das Teilproblem auf Systemebene mithilfe des genetischen Algorithmus optimiert wird und auf Komponentenebene ein Matlab-Algorithmus für *sequentielle quadratische Programmierung* zum Einsatz kommt.

In der Arbeit von [Joung und Do Noh, 2014] soll das Problem des effizienten Be- und Entladens eines gegebenen Containers mit frei geformten dreidimensionalen Objekten gelöst werden, wobei die geometrische Form, die Be- und Entladefreundlichkeit und die Packungsdichte berücksichtigt werden. Ausgehend von einem Szenario, in dem Arbeitskräfte Transportgut in zusammengestellten Einheiten einpacken, stellen die Autoren einen Ansatz vor, der sowohl auf der Analyse von Arbeitsmustern beim Laden von LKWs als auch auf bereits bekannten Gruppierungsmethoden aus dem Fertigungssektor basiert. Ziel der Untersuchungen ist die Effizienzsteigerung der Lagerung im Fertigungssektor. Die entwickelte Methode besteht aus einem Gruppierungsalgorithmus, einem Sequenzierungsalgorithmus, einem Orientierungsalgorithmus und einem Ladealgorithmus. Diese Algorithmen berücksichtigen die Ladereihenfolge, die Orientierung des Transportguts, die Kollisionsfreiheit während und nach der Be- bzw. Entladung und die Positionierung des Transportguts, unterstützen eine Effizienzprüfung über die Berechnung der Arbeitsleistung und ermöglichen eine Be- und Entladesimulation.

[Wu et al., 2017] untersuchen ein Szenario im industriellen Bereich, in dem Kunden drei unterschiedliche Arten von Verbindungselementen bestellen. Das Gesamtproblem zielt auf die maximale Auslastung eines Containers ab und umfasst drei Teilprobleme, welche in drei Optimierungsstufen gelöst werden. In der ersten Stufe werden die Verbindungselemente ihrem Typ

nach geordnet und in kleine Kartons verpackt, in der zweiten Stufe werden die Kartons für eine Erleichterung des Transports in Kisten gepackt und in der letzten Stufe werden die Kisten in einen Container verladen. Die Autoren schlagen für jede der drei Optimierungsstufen einen heuristischen Algorithmus zur Lösung des jeweiligen Teilproblems vor. Die Verbindungselemente werden mittels einer Routine in Kartons variabler Größe gepackt, wobei alle Arten von Kartons in jeder Richtung ausprobiert werden und für die Verbindungselemente eine aus zwei oder acht (abhängig vom gewählten Typ) vordefinierten Verpackungsstrategien angewendet wird. Die Platzierung der Kartons erfolgt mithilfe eines erweiterten heuristischen *Blockladealgorithmus*, der auf einer *Multilayer*-Suchstrategie aus [D. Zhang et al., 2012] basiert. Die Containerbeladung dagegen wird mithilfe des originalen *Multilayer*-Algorithmus berechnet. Aus Sicht der Autoren arbeitet der vorgeschlagene Algorithmus effizient und liefert zufriedenstellende Ergebnisse. Diese Aussage wurde durch von realen Fällen inspirierte Szenarien validiert.

Im Beitrag von [Ma et al., 2018] sollen die allgemeinsten Formen von Packingproblemen mit unregelmäßigen Formen im dreidimensionalen Raum betrachtete werden, bei denen sowohl die Behälter als auch die Objekte beliebige Formen annehmen können und kontinuierliche Rotationen der Objekte erlaubt sind. Die Autoren schlagen eine heuristische Methode vor, in der ein kontinuierliches Optimierungsverfahren mit einem kombinatorischen Optimierungsverfahren kombiniert werden. Ausgehend von einer initialen Platzierung einer vorgegebenen Anzahl von Objekten werden die Positionen und Ausrichtungen der Objekte mittels kontinuierlicher Optimierung berechnet. Diese besteht aus zwei Phasen, dem Schrumpfen von Objekten auf eine geringe Größe und der iterativen Anpassung der Objektkonfiguration, einschließlich ihrer Größe, Position und Ausrichtung, während sie allmählich auf ihre ursprüngliche Größe anwachsen. Den Autoren zufolge erzielt die Verwendung eines solchen iterativen Relaxationsschemas zur Vermeidung von Kollisionen ein relativ gutes Ergebnis für das Packingproblem. Für eine weitere Verbesserung der Packungsdichte werden bei der anschließenden kombinatorischen Optimierung die Lücken zwischen den Objekten verringert, indem die eingesetzten Objekte untereinander ausgetauscht oder durch andere ersetzt sowie neue Objekte eingefügt werden.

[Lutters et al., 2012] befassen sich mit dem Packing von Objekten komplexer Form im Bereich der additiven Fertigung. Der vorgeschlagene heuristische Ansatz nutzt den sogenannten *Paranuss*-Effekt zur Maximierung des Füllungsgrads der Produktionschargen. Der Algorithmus besteht aus einem iterativen Prozess, bei dem in einem ersten Schritt komplexe Formen durch ein Hüllvolumen mit zunehmender Komplexität in jedem Iterationsschritt repräsentiert werden, und einer anschließenden Positionierung und Ausrichtung der Objekte durch eine Simulation der Behältervibration. Neben der Packungsdichte wird auch die Oberflächenqualität als Bewertungskriterium herangezogen. Nach Ansicht der Autoren verbessert die Nutzung des *Paranuss*-Effekts die Qualität der Lösung von Packingproblemen mit hoher Dichte.

Ein weiterer heuristischer Ansatz wird in [X. Liu et al., 2015] vorgestellt. Die Autoren schlagen einen konstruktiven Algorithmus unter dem Namen *HAPE3D* vor, der auf dem Prinzip der minimalen potenziellen Energie beruht. Ziel bei dem vorgestellten Ansatz ist, beliebig geformte Bauteile in einem kastenförmigen Behälter so anzuordnen, dass dessen Höhe minimiert wird. Der heuristische Algorithmus zur Positionierung und Drehung der Polyeder berücksichtigt für jeden Polyeder bis zu acht Drehwinkel um jede Achse. Nach Meinung der Autoren ist *HAPE3D* sehr vielversprechend für die Lösung des vorgestellten dreidimensionalen Packingproblems. Um die Lösungsqualität zu verbessern, verwenden die Autoren außerdem eine genetische Optimierung als Erweiterung, wodurch der Hybridalgorithmus *HAPE3D+SA* entsteht.

In dem Beitrag von [Rodrigues et al., 2013a] und [Rodrigues et al., 2013b] wird ein hybrider evolutionärer Algorithmus zur Lösung von Layoutproblemen in der Architektur vorgestellt. Der Algorithmus kombiniert eine globale evolutionäre Strategie mit der lokalen stochastischen

Hill-Climbing-Technik und soll im Sinne eines hybriden Ansatzes Vorteile aus beiden Verfahren nutzen. Eingesetzt werden soll der entwickelte Algorithmus für die Positionierung von rechteckigen Räumen sowie deren Türen und Fenstern. Als Nebenbedingungen können topologische Präferenzen wie Nachbarschaften vorgegeben oder ganze Cluster für gemeinsame funktionale Raumeinheiten definiert werden.

3.5 Ganzheitliche Ansätze und die Einbettung im Gesamtentwurf

Wie in der Einleitung angesprochen steht die Forschung im Bereich der Konfiguration und Integration von Systemen vor der großen Herausforderung, effiziente Frameworks zur Entwurfsautomatisierung zu schaffen, welche alle gekoppelten Aspekte von Entwurfsprozessen auf Systemebene ganzheitlich abbilden können. Obwohl sich die meisten Forschungsarbeiten mit speziellen Problemstellungen befassen, existieren auch Ideen und Ansätze für die allgemeine problemunabhängige Unterstützung von Integrations-, Konfigurations- und Packingprozessen.

Wie etwa in der Arbeit von [Cagan et al., 2002], in der die Autoren bereits im Jahr 2002 die Notwendigkeit für die Schaffung eines allgemeinen Frameworks erkennen. Die Autoren schlagen den theoretischen Aufbau eines Frameworks vor und nennen die Optimierungsstrategie, die geometrische Repräsentation und die daraus resultierende Interferenzbewertung als mögliche generische Bestandteile eines Layouttools, bleiben jedoch eine konkrete ganzheitliche und problemunabhängige Implementierung schuldig.

[Fadel und Wiecek, 2015] nennen in diesem Zusammenhang das Problem, dass im klassischen Systementwurf die Systemintegration der Einzelkomponentenauslegung folgt, was einem ganzheitlichen Ansatz im Wege steht. Die Autoren argumentieren, dass die Veränderung der Bauteilform parallel zu der Bauteilpositionierung eindeutig zu weitaus besseren Packinglösungen führen kann. Als Folge ist einer der Autoren an den Arbeiten von [Dong et al., 2011] und [Tiwari et al., 2014] beteiligt, in denen ein gleichzeitiger Ansatz für die Formgebung und die Positionierung sowie Orientierung mithilfe eines Morphing-Verfahrens verfolgt wird. Neben den geometrischen werden im Sinne eines ganzheitlichen Ansatzes auch physikalische Randbedingungen betrachtet. In der Studie von [Dandurand et al., 2014] wird die optimale Platzierung von sechs Komponenten im Motorraum eines Hybrid-Elektrofahrzeugs durchgeführt, wobei eine der Komponenten unter anspruchsvollen thermischen Kriterien konstruiert wird. Die Arbeiten zeigen im Automobilbereich, dass es durchaus möglich sein könnte allgemeine komplexe Packingprobleme ganzheitlich zu lösen.

[Panesar et al., 2015] stellen ein Framework vor, das den Entwurf von additiv gefertigten Multimaterialbauteilen mit eingebetteten Funktionssystemen unterstützt. Das entwickelte Framework hat drei Hauptbestandteile, die Platzierung von Komponenten innerhalb eines Gesamtproduktes, das Routing zwischen diesen Komponenten und die Berücksichtigung der Auswirkungen der Integration dieser Komponenten auf die strukturelle Reaktion des Gesamtproduktes durch die Modifikation der Struktur mithilfe einer Optimierungsstrategie. Übergeordnet zu diesen drei Bestandteilen sind Kopplungsstrategien umgesetzt, welche Wechselwirkungen zwischen den Bestandteilen berücksichtigen. Vervollständigt wird die Entwurfsstrategie durch die Einbeziehung geometrischer, fertigungstechnischer und anwendungsbezogener Entwurfsrestriktionen und Strategien, die für eine effiziente und flexible Berechnung der Ergebnisse eingesetzt werden. Insgesamt kommt der Ansatz einer umfassenderen Packingstrategie nahe, allerdings ist er auf Multimaterialbauteile und die Additive Fertigung begrenzt und nicht auf allgemeinere Packingaufgaben anwendbar.

Im Bereich der Raumfahrt stellen [Giorgio Fasano et al., 2014] und [Giorgio Fasano et al., 2015] einen ganzheitlichen Ansatz zur Auslegung eines *Automated Transfer Vehicle* vor. Dieser implementiert eine Gesamtarchitektur, die auf einer mathematischen Bibliothek basiert, welche den Kern des gesamten Optimierungsframeworks darstellt. Die Gesamtauslegung wird auf Teilprobleme aufgespalten, welche durch entsprechende mathematische Bibliotheksmodule, bestehend aus spezifischen mathematischen Modellen und heuristischen Algorithmen, repräsentiert werden. Diese werden iterativ gelöst und durch ein übergreifendes System Management Modul verwaltet. Sowohl Vorwärts- als auch Rückwärtsiterationen sind bei dem iterativen Lösungsprozess vorgesehen, wobei ein rekursiver Prozess durchgeführt wird, bis die gewünschte Lösung mit den gegebenen statischen und dynamischen Gleichgewichtsanforderungen auf Systemebene (also aus Sicht des Gesamtproblems) konform ist. Außerdem ist eine grafische Oberfläche integriert, um menschliche Interaktion zu ermöglichen, da festgestellt wurde, dass einige auf menschlicher Wahrnehmung basierende Bewertungskriterien kaum von mathematischen Modellen berücksichtigt werden können [Giorgio Fasano et al., 2014].

[Yakovlev et al., 2018] präsentieren ein domänenunabhängiges informationsanalytisches Modell für die Synthese optimaler Systemkonfigurationen zur Unterstützung und Entscheidungsfindung bei der Integration von Systemkomponenten mit komplexer räumlicher Form. Die Autoren schlagen ein auf Basis von allgemeinen Freiheitsgraden und Bewertungskriterien aufbauendes Modell vor, welches zur Beschreibung der benötigten Datenstrukturen für die Entwicklung des Entwurfsraumes verwendet werden kann. Dafür wird eine Hierarchie von Klassen für die Verarbeitung, Transformation und Anpassung von Daten zur Bildung zulässiger lokal optimaler Konfigurationen entwickelt. Die Ausführung konkreter Anwendungsbeispiele bleibt in der Veröffentlichung allerdings aus.

Die Forschungsarbeiten von [Jessee et al., 2020] befassen sich mit einem Ansatz, der sowohl die Konfiguration von Geräten als auch der räumlichen Ausprägung ihrer Vernetzungspfade berücksichtigt. Für die gleichzeitige Optimierung werden gradientenbasierte Verfahren und Topologie-Optimierungstechniken verwendet. Zusätzlich zu geometrischen Nebenbedingungen unterstützt die vorgestellte Methode eine Optimierung auf der Grundlage des Systemverhaltens, indem physikalische Ziele und Einschränkungen einbezogen werden. In den Beispielen werden mehrere physikalische Domänen mithilfe von pauschalen Parametern oder Finite-Elemente-Modellen repräsentiert.

[Peddada et al., 2021a] stellen ein zweistufiges sequentielles Entwurfs-Framework vor, um eine gleichzeitige physikbasierte Packing- und Routing-Optimierung durchzuführen. In der Stufe werden störungsfreie Ausgangslayouts generiert, die in der zweiten Stufe als Startpunkte für eine kontinuierliche physikbasierte Optimierung dienen. Für die Generierung der Ausgangslayouts werden drei verschiedene Strategien vorgestellt, die sog. *Force-Directed Layout* Methode, eine Erweiterung des *Shortest Path* Algorithmus und ein sog. *Unique Geometric Topology* Generierungsalgorithmus. Für die anschließende Optimierung wird ein gradientenbasiertes Topologieoptimierungsverfahren verwendet, um gleichzeitig sowohl die Position der Komponenten als auch die Wege der Zwischenverbindungen zu optimieren. Neben den geometrischen Randbedingungen wird auch das Systemverhalten in die Optimierung einbezogen, indem physikalische Ziele und Einschränkungen berücksichtigt werden. In den Augen der Autoren lässt sich sagen, dass das vorgestellte Framework für die Entwurfsautomatisierung mehrere Elemente integriert, die einen Schritt hin zu einer umfassenderen Lösung von dreidimensionalen Packing- und Routing-Problemen unter Berücksichtigung sowohl geometrischer als auch physikalischer Aspekte darstellen. Wie die Integration in einen bestehenden Entwurfsprozess stattfinden könnte, wird nicht näher diskutiert.

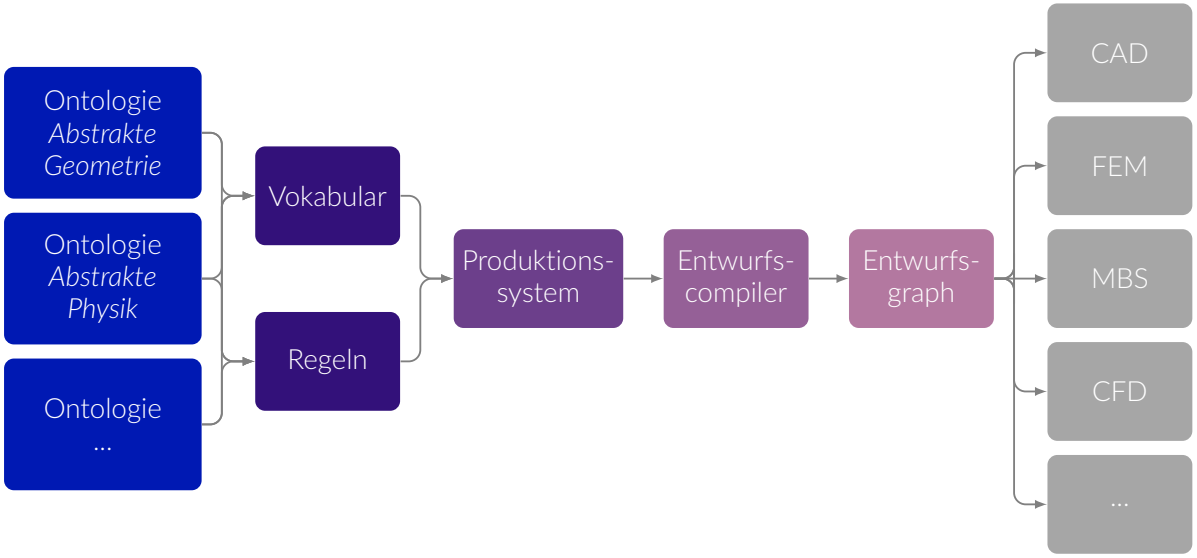


Abbildung 3.1: Architektur von graphenbasierten Entwurfssprachen.

3.6 Graphenbasierte Entwurfssprachen

Wie eingangs beschrieben, wird das in der vorliegenden Arbeit entwickelte Framework zur Lösung von Layout- und Packingaufgaben auf Basis der sog. graphenbasierten Entwurfssprachen implementiert. Zur Modellierung und Ausführung der genannten Entwurfssprachen wird die Software *Design Cockpit 43*® [IILS, 2022] verwendet, welche an der *IILS mbH* [1] entstand und bis heute weiterentwickelt wird. Der mit den Entwurfssprachen einhergehende Entwurfsansatz lässt sich am besten mit der Technologie von Programmiersprachen und deren Compilern vergleichen. Graphenbasierte Entwurfssprachen entsprechen dabei dem Quellcode, in dem das gesamte Entwurfswissen gespeichert wird [J. Schmidt, 2017]. Das *Design Cockpit 43*® entspricht hierbei dem Compiler, in dem die Designsprache kompiliert. Während das Ergebnis im klassischen Fall der Programmiersprachen ein lauffähiges Computerprogramm ist, ist das Ergebnis bei der Verwendung von Entwurfssprachen und des *Design Cockpit 43*® die Entwurfslösung eines gewünschten Systems [J. Schmidt, 2017].

Inspiriert sind die graphenbasierten Entwurfssprachen ursprünglich von natürlichen Sprachen, in denen es seit jeher möglich ist, alle Arten von Informationen über Systeme, Domänen und Detailgrad hinweg auszutauschen. Genauso wie das Vokabular natürlicher Sprachen zusammen mit einem Regelwerk zu dessen Anwendung eine valide Grammatik bildet, soll das Entwurfsvokabular mit einem zugehörigen Regelwerk einen validen Entwurfsraum von Ingenieursystemen repräsentieren. Während jede regelkonforme Kombination der Sprachbausteine in natürlichen Sprachen zu grammatikalisch korrekten Aussagen führt, führt sie in Entwurfssprachen zu gültigen Entwurfsmodellen. In diesem Sinn bestehen graphenbasierte Entwurfssprachen, wie sie in Abb. 3.1 dargestellt sind, aus einem Vokabular für die Modellbildung und einem Satz von Regeln zur Modelltransformation, sowie aus einer ausführbaren Regelsequenz, welche in ihrer Gesamtheit als Produktionssystem bezeichnet wird. Wird das *Design Cockpit 43*®zur Programmierung von Entwurfssprachen verwendet, stehen dem/der Benutzer/-in zur Ontologiemodellierung Klassendiagramme, zur Modellierung der Regeln graphische Regeln sowie in Java kodierte sog. Javarules und zur Modellierung des Produktionssystems Aktivitätsdiagramme sowie Subprogramme zu deren Strukturierung zur Verfügung. Das Vokabular kann

[1] IILS Ingenieurgesellschaft für Intelligente Lösungen und Systeme mbH, https://www.iils.de/

entweder neu formuliert werden oder es wird aus bereits zuvor erstellten Ontologien gewonnen. Die Entwurfssprachen müssen bis hin zum Produktionssystem manuell programmiert werden und können anschließend automatisch in einer Übersetzungsmaschine, im konkreten Fall dem *Design Cockpit 43*®ausgeführt werden. Der regelbasierte Ausführungsmechanismus instanziiert ein zentrales abstraktes Produktmodell in Form eines Entwurfsgraphen. Ein Graph eignet sich zur Repräsentation von Entwurfsmodellen deshalb besonders gut, weil er jeden beliebigen Ausdruck eines Systems als Knoten sowie die Abhängigkeiten zwischen den einzelnen Teilsystemen uneingeschränkt als Kanten darstellen kann. Somit erlaubt er eine Entwurfsbeschreibung, welche alle gekoppelten physikalisch-funktionalen Eigenschaften eines Systems repräsentiert. Die automatisierte Ausführbarkeit von Entwurfssprachen ermöglicht eine computergestützte Exploration von Entwurfsräumen, in denen jede Produktvariante als zusammenhängender Entwurfsgraph vorliegt. Nach der Generierung des Entwurfsgraphen können über Plugins, welche den Graphen analysieren, automatisch domänenspezifische Entwurfsmodelle in die domänenspezifischen Sprachen externer Software übersetzt werden. Werden die Modelle in der jeweiligen Software simuliert, können die Ergebnisse aus deren Berechnungen in den Entwurfsgraphen zurückgeführt werden, um die Entwurfsschleife für die Optimierung der generierten Entwurfsmodelle zu schließen, welche schließlich zur Entscheidungsfindung führt.

KAPITEL 4

Ein Framework für Layout und Packing in graphenbasierten Entwurfssprachen

Die Entwicklung eines Frameworks auf Basis eines fundamentalen, ganzheitlichen und systemorientierten Ansatzes, um die automatisierte Lösung von Layout- und Packingproblemen mithilfe von systematischen, flexiblen und wissenschaftlich fundierten Entwurfsmethoden zu unterstützen, welches für verschiedenste technische Branchen die Möglichkeit bietet, durch eine effiziente Generierung und Bewertung von Entwurfslösungen neue Konfigurationen zu erforschen und dadurch fortschrittlichere und komplexere Systeme zu realisieren, um letztlich die Steigerung der Energieeffizienz, die Reduktion von Emissionen und die Stärkung der wirtschaftlichen Wettbewerbsfähigkeit zu ermöglichen. Mit diesem Ziel vor Augen beginnt die Entwicklung einer Vorgehensweise zur Schaffung eines solchen Frameworks. Die in *2.2 Wissenschaftliche Motivation und Lösungsansatz* geführte Diskussion legt die zentralen Grundsteine dafür, im weiteren Verlauf des Kapitels soll nun das Ergebnis präsentiert werden. Dabei wird ein in graphenbasierten Entwurfssprachen umgesetztes, automatisiertes und flexibles Framework zur Lösung von Layout- und Packingaufgaben vorgestellt, welches problem- und systemangepasste Lösungsstrategien ermöglicht und gleichzeitig auch allen weiteren in der oben genannten Diskussion bereits abgeleiteten notwendigen Forderungen genügt.

Das vorliegende Kapitel führt den Leser hierzu beginnend mit der allgemeinen Architektur des Frameworks in *4.1 Architektur des Frameworks* durch dessen einzelne Bestandteile, wobei im Sinne einer Vorverarbeitung in *4.2 Entwurfssprache für Geometrieapproximation* zunächst erläutert wird, wie Geometrien eingelesen und für die weitere Verarbeitung vereinfacht werden können. Das Kapitel setzt in *4.3 Optimierung im Packing Framework* mit der entwickelten Ontologie für die abstrakte Optimierung fort, welche als zusätzliches Element die umfassenden Entwurfssprachenontologien erweitern kann und schließt in *4.4 Entwurfssprache für Packingprobleme* mit der auf den ersten beiden Entwurfsspachen aufbauenden Entwurfssprache zur Modellierung und Ausführung von Layout- und Packingproblemen ab.

4.1 Architektur des Frameworks

Der Kern des entwickelten Frameworks umfasst zunächst die Repräsentation aller, den Layout- und Packingprozessen innewohnender Elemente. Diese reichen von den möglichen Freiheitsgraden und der räumlichen Diskretisierung über die Geometrien der einzelnen beteiligten Komponenten bis hin zu den Rand- und Nebenbedingungen, welche von mathematischer, physi-

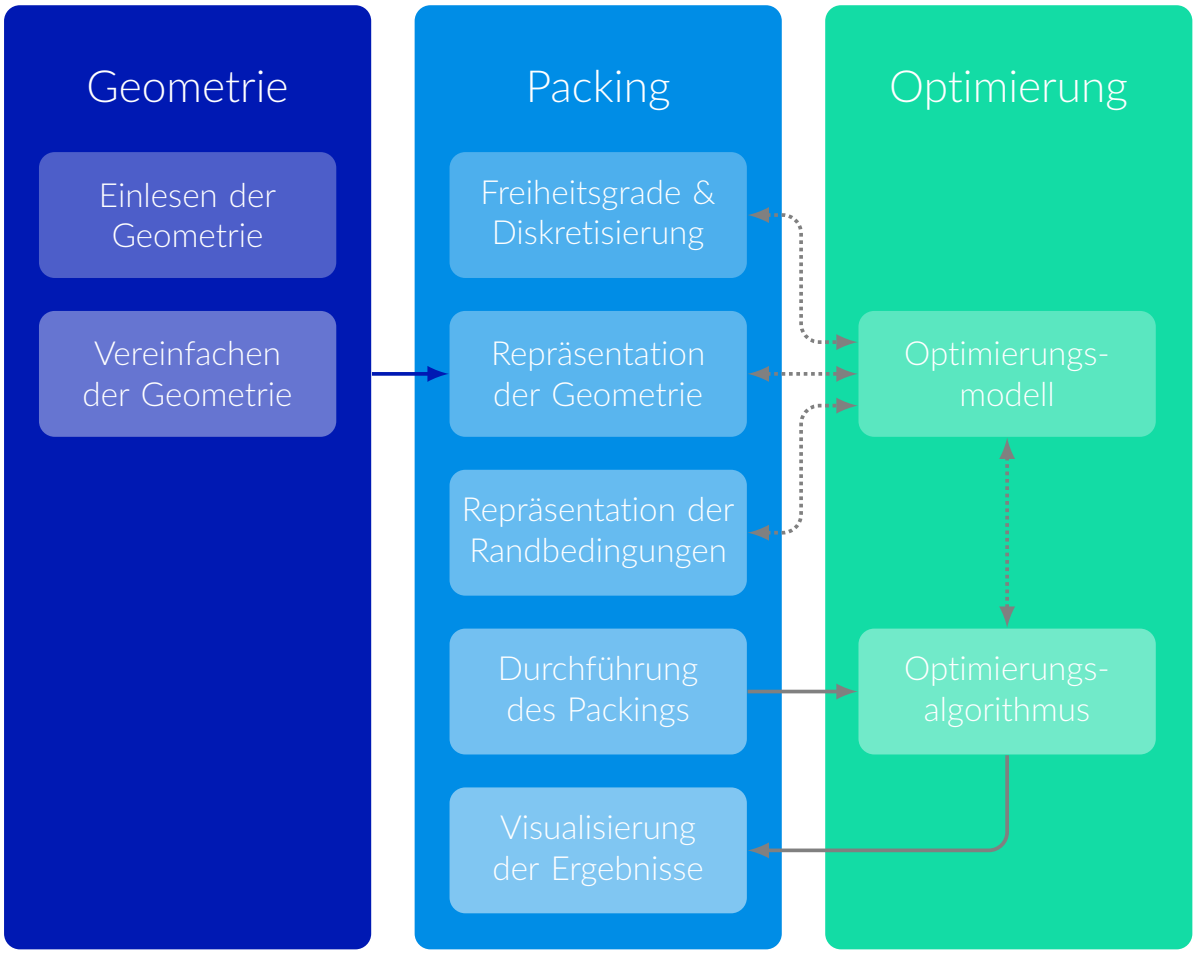


Abbildung 4.1: Die drei Säulen des Packingframeworks.

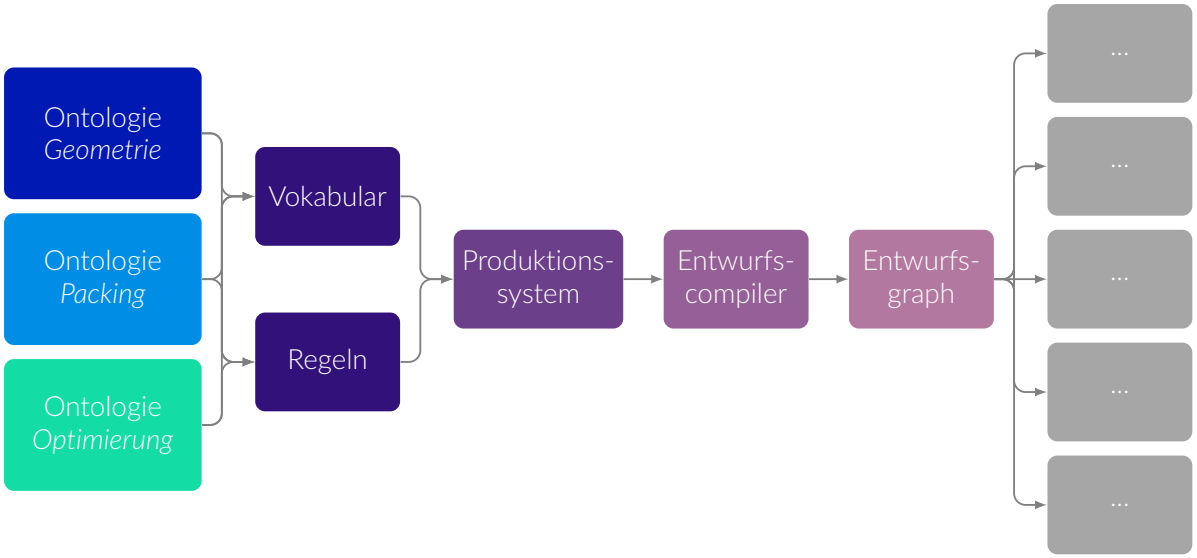


Abbildung 4.2: Ontologien für Geometrieapproximation, Packing und Optimierung.

kalischer oder geometrischer Natur sein können. Für diesen Teil wurde die Entwurfssprache für Layout- und Packingprobleme entwickelt, welche als zentrale Säule in Abb. 4.1 dargestellt ist. Die Repräsentation dieser Elemente bildet die Basis für die digitale Modellierung und ganzheitliche Darstellung von packingspezifischen Problemformulierungen durch den/die Anwender/-in. Die Entwurfssprache unterstützt den Aufbau des Gesamtproblems durch eine Reihe vordefinierter Modelle, welche die einzelnen Elemente wie Freiheitsgrade, Diskretisierung, Komponentengeometrie oder Nebenbedingungen repräsentieren. Daneben werden Funktionalitäten für die Durchführung des automatisierten Packingprozesses zur Verfügung gestellt. Auch die Visualisierung der Ergebnisse kann in der beschriebenen Entwurfssprache durchgeführt werden, indem die Ergebnisse des Packingprozesses ausgelesen und die errechneten Werte für die Freiheitsgrade auf die Positionierung der jeweiligen Komponentengeometrien angewandt werden. Damit erhält der/die Anwender/-in neben der Ausgabe der berechneten Lösung als Werteliste zusätzlich eine visuelle Darstellung der Ergebnisse.

Wie in Abschnitt 2.2 bereits angesprochen wurde, kann der Prozess der Konfiguration und des Positionierens von Systemkomponenten automatisiert werden, wenn das Packingproblem als Optimierungsproblem formuliert wird, das automatisiert gelöst werden kann. Das bedeutet, dass alle packingspezifischen Elemente gleichzeitig Elemente einer Optimierung sind, dass also den Freiheitsgraden entsprechende Optimierungsvariablen und den Geometrien und Nebenbedingungen entsprechende Optimierungsmodelle beispielsweise in mathematischer Form von Gleichungs- und Ungleichungssystemen vorliegen müssen. Theoretisch ergeben sich die Variablen und -beschränkungen für die Optimierung ganz natürlich aus der Wahl der Entwurfsfreiheitsgrade, der Formulierung des Lösungsraums, der Repräsentation der Geometrien, sowie der packingspezifischen Entwurfsbeschränkungen und Zielfunktionen. Für die Modellierung und Lösung der als Optimierungsaufgaben zu definierenden Packingprobleme wurde die Entwurfssprache zur Optimierung entwickelt. Diese ist durch die rechte Säule in Abb. 4.1 repräsentiert. Da die Ausführung von Optimierungen nicht ausschließlich auf Layout- und Packingfragestellungen anwendbar sein soll, ist diese Entwurfssprache als separates und autonomes Modul umgesetzt. Dieses wurde zwar aus der Notwendigkeit für ein Packingframework heraus geboren, kann aber für allgemeine Optimierungsaufgaben verwendet werden. Es beinhaltet Methoden zur Formulierung von Optimierungsproblemen sowie eine Reihe von Funktionen, um unterschiedliche Optimierungsverfahren anzusprechen.

Für die Modellierung und Berechnung von Layout- und Packingaufgaben reichen grundsätzlich die beiden bisher beschriebenen Entwurfssprachen aus. Als Erweiterung wurde noch eine dritte Entwurfssprache entwickelt, welche zum einen die Anwendung vereinfachen soll und zum anderen eine Anpassung der Komplexität der Packingaufgaben hinsichtlich der Geometrierepräsentation unterstützen soll. Konkret ermöglicht die in Abb. 4.1 links dargestellte Entwurfssprache für Geometrieapproximation das Einlesen von Geometrien aus beliebigen Datenformaten und bietet eine Reihe an geometrischen Vereinfachungen, welche in einem Vorverarbeitungsschritt aufgerufen werden können, um eine geeignete und problemangepasste Repräsentation der Packinggeometrien zu erhalten. Alle drei Bestandteile des Packingframeworks werden in Form von Entwurfssprachenplugins implementiert. Das heißt, dass jede der drei Säulen in der vorliegenden Arbeit als eigenständige Ontologie modelliert wird, welche das Repertoire der bestehenden Basisontologien in graphenbasierten Entwurfssprachen erweitert, wie auch Abb. 4.2 entnommen werden kann.

4.2 Entwurfssprache für Geometrieapproximation

Soll ein Framework für Packingprozesse geschaffen werden, muss zunächst über eine Möglichkeit nachgedacht werden, wie bestehende Komponenten und deren Geometrien eingelesen und verarbeitet werden können. Dabei muss berücksichtigt werden, dass die Komponentengeometrien in den meisten Fällen in unterschiedlichen Datenformaten vorliegen und erst einmal in eine geeignetes internes Format transformiert werden müssen. Den meisten Datenformaten ist die klassische digitale Repräsentationsform über ein Oberflächennetz gemein. Diese Darstellung ist für Packingprozesse allerdings häufig von Nachteil, da sie unter anderem durch eine potenzielle Nichtkonvexität zu beliebig komplizierten Modellen für die Packingoptimierung führt. In der vorliegenden Arbeiten wurde daher die Entwurfssprache für Geometrieapproximation geschaffen, welche zum einen den Import von beliebigen Datenformaten unterstützt und zum anderen eine Vereinfachung der Geometrien für effizientere Packingprozesse ermöglicht. Angelehnt an den realen Workflow für den/die Anwender/-in beginnt das vorliegende Kapitel mit einer Erläuterung des Geometrieimports in *4.2.1 Import von Geometrie*, gefolgt von der Vorstellung der entwickelten und implementierten Bibliothek für die Geometrievereinfachung in *4.2.2 Approximation von Geometrie*, wo auch ausführlich auf die einzelnen sog. *Geometrieapproximationen* eingegangen wird.

4.2.1 Import von Geometrie

Für den Import der Geometrie wird auf einen Mechanismus der Software *Design Cockpit 43*®[J. Schmidt, 2017] zugegriffen, welcher das eingelesene Datenformat in ein trianguliertes Oberflächennetz in Form eines *vtkPolyData* Objektes aus der frei verfügbaren Bibliothek *The Visualization Toolkit* (VTK) [Schroeder et al., 2006] konvertiert. Eingelesen werden können die Formate:

- .vtk or .vtp
- .step or .stp
- .stl
- .obj
- .3ds
- .brep

Das *vtkPolyData* Objekt wird im Anschluss mithilfe der in der vorliegenden Arbeit entwickelten Entwurfssprache in ein internes `GAMesh` Format umgewandelt. Das `GAMesh` Format bildet dann die Basis für die weitere Verarbeitung wie beispielsweise für die Erstellung von vereinfachten geometrischen Ersatzrepräsentationen. Diese sollen im folgenden Kapitel vorgestellt werden.

4.2.2 Approximation von Geometrie

Ziel der im vorliegenden Kapitel vorgestellten Entwurfssprache ist die Bereitstellung einer umfangreichen Auswahl an Geometrieannäherungen. Diese Ersatzgeometrien sollen im Packingprozess anstelle der Originalgeometrien verarbeitet werden und dienen der Effizienzsteigerung bei der Optimierung des Packings. Je nach Charakter des Packingproblems eignen sich bestimmte geometrische Darstellungen besser als andere, bei der Modellierung müssen also die

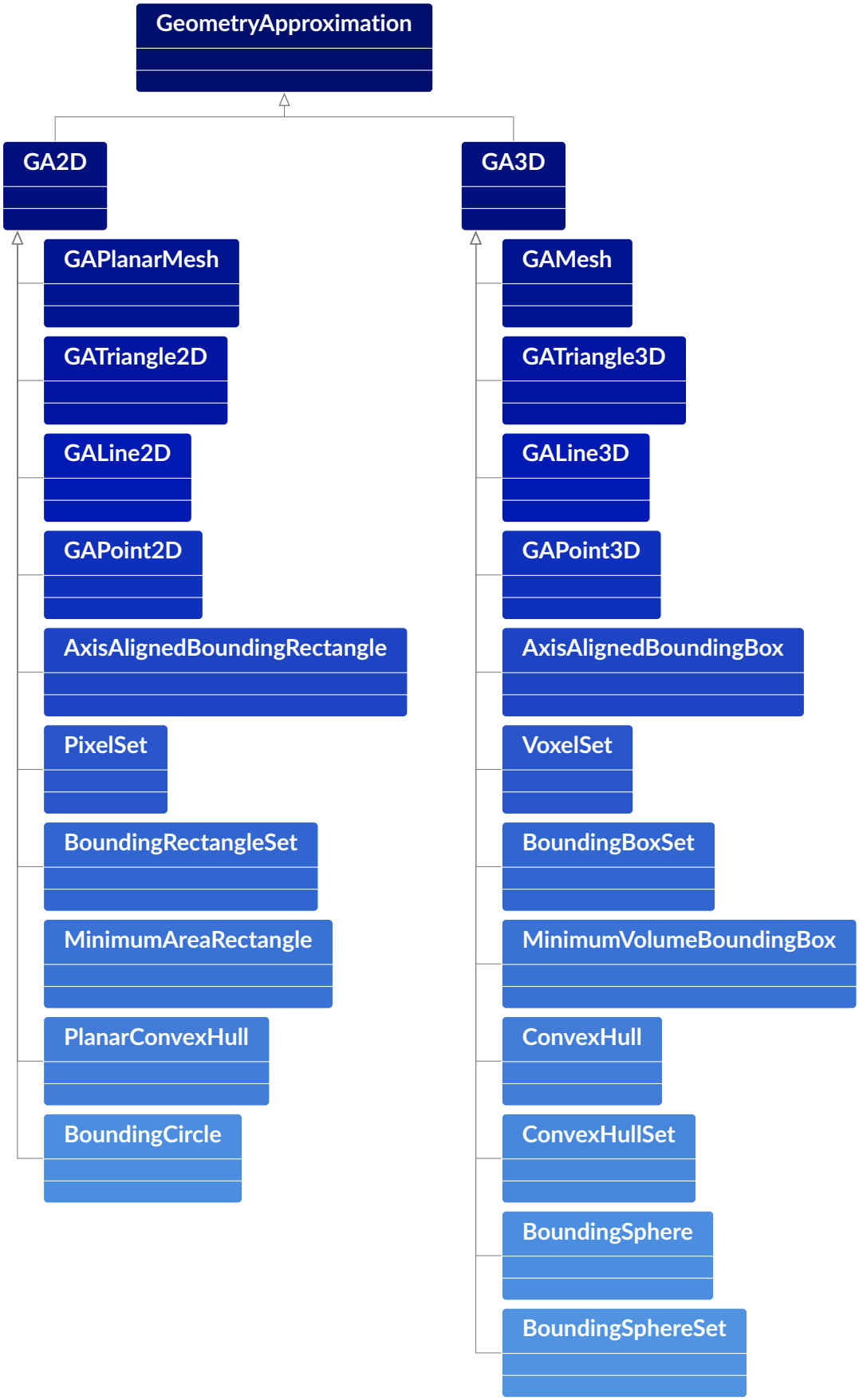
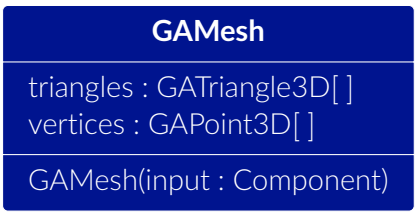


Abbildung 4.3: Ontologie für zwei- und dreidimensionale Geometrieapproximationen.

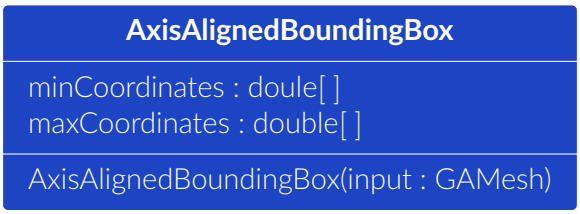
Geometrieapproximationen und das Packingmodell aufeinander abgestimmt werden. Welche Geomeriemodelle hierbei zu welchen Packingmodellen passen, soll dem Leser im Laufe der vorliegenden Arbeit nach und nach erläutert werden. Im Folgenden wird zunächst die entwickelte und implementierte Ontologie zur Geometrieapproximation vorgestellt und die einzelnen Methoden im Detail beschrieben. Es sei an dieser Stelle darauf hingewiesen, dass die im Folgenden vorgestellten Algorithmen für einige der internen Berechnungen ebenfalls auf Funktionalitäten der frei verfügbaren Bibliothek *The Visualization Toolkit* (VTK) [Schroeder et al., 2006] zurückgreifen. Die entwickelte Ontologie kann Abb. 4.3 entnommen werden. Wie der Abbildung zu entnehmen ist, findet eine Klassifizierung in zweidimensionale und dreidimensionale Geometrieapproximationen statt. Für beide Varianten existiert eine Auswahl an möglichen Vereinfachungen, auf die nachfolgend separat eingegangen werden soll.

Interne Oberflächentriangulation



Die Klasse `GAMesh` steht für *Geometry Approximation Mesh* und wird für die interne Repräsentation aller dreidimensionaler Packinggeometrien als trianguliertes Netz verwendet. Übergabeparameter für den Konstruktor ist die entwurfssprachenineterne abstrakte Geometrierepräsentation (Klasse `Component`). `GAMesh` greift für die Triangulation auf die Klassen `GATriangle3D`, `GALine3D` und `GAPoint3D` zu. Diese Darstellung bildet die Basis für alle im folgenden beschriebenen Geometrievereinfachungen. Für die Repräsentation in zwei Dimensionen werden die Klassen `GAMesh2D`, `GATriangle2D`, `GALine2D` und `GAPoint2D` verwendet. Die Klassen `GATriangle3D` und `GATriangle2D` werden außerdem im Packing benötigt, um Zwänge zwischen Oberflächen zu modellieren. Die Klassen `GALine2D` und `GALine3D` sowie `GALine2D` und `GALine3D` finden ebenfalls im Packing Anwendung, um Randbedingungen zwischen Kanten oder Punkten von Geometrien abzubilden.

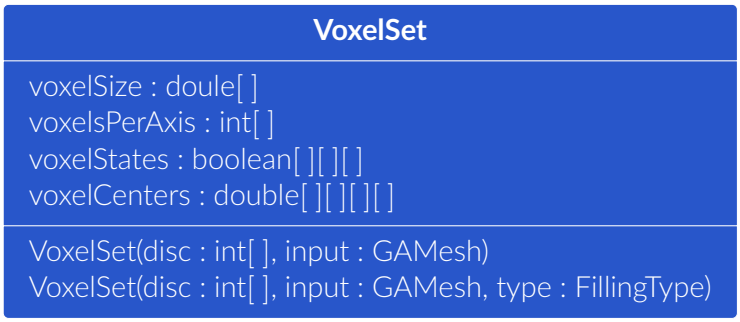
Achsenorientierter Hüllquader



Die `AxisAlignedBoundingBox` Klasse ist für die Erstellung eines achsenorientierten Hüllquaders in drei Dimensionen implementiert. Übergabeparameter ist ein dreidimensionales trianguliertes Oberflächennetz in Form eines `GAMesh` Objektes. Bei der Erzeugung findet eine Iteration über alle Eckpunkte des Netzes statt, wobei die globalen minimalen und maximalen Koordinaten zwischengespeichert werden. Aus diesen lassen sich anschließend die Eckpunkte des Quaders berechnen. Aufgrund der grundsätzlichen Konvexität von Quadergeometrien ist es hinreichend, ausschließlich die Eckpunkte des Netzes zu berücksichtigen, die Kanten des Netzes können dabei vernachlässigt werden.

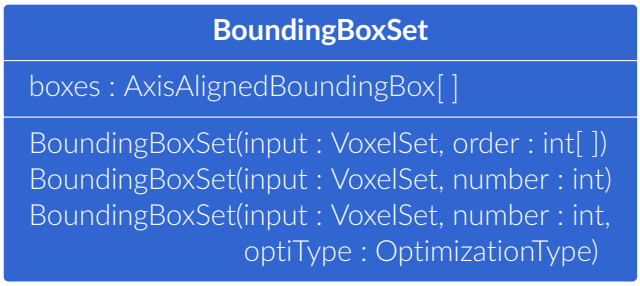
Voxelisierung

Die Klasse `VoxelSet` dient zur Erstellung einer Voxelisierung einer dreidimensionalen Geometrie. Die verwendeten Übergabeparameter sind eine dreidimensionale Oberflächentriangulation in Form eines `GAMesh` Objektes, die Diskretisierung der Voxelisierung in alle Raumrichtungen sowie optional die Berechnungsart der Voxelfüllung (sog. Voxelstatus) in Form einer Enumeration. Hierbei stehen die Enumerationswerte FillingType.CENTER, FillingType.CORNERS, FillingType.INSIDERECTANGULAR, FillingType.SQUAREDDISTANCE sowie für eine besonders



präzise Überprüfung FillingType.ByCollision zur Auswahl. In einem ersten Schritt wird ein dreidimensionales Raster gemäß der vorgegebenen Diskretisierung erzeugt, welches die Koordinaten für die Mittelpunkte der einzelnen Voxel vorgibt. In einer anschließenden geschachtelten Iteration über die Dreiecke der Triangulation und über das Raster wird berechnet, ob die Voxel gefüllt oder leer sind, was abschließend in der Voxelisierung abgespeichert wird. Wird FillingType.Center als Enumerationswert übergeben, wird für den Füllungsstatus lediglich geprüft, ob der Mittelpunkt des Voxels innerhalb der Geometrietriangulation liegt. Diese Variante führt zu schnellen Ergebnissen, führt jedoch mit steigender Voxelgröße zu hohen Ungenauigkeiten. Bei Übergabe von FillingType.Corners werden die Eckpunkte des Voxels überprüft und führen zu gefülltem Status, wenn mindestens einer der Eckpunkte in der Geometrietriangulation enthalten ist. Eine weitere Variante ist durch FillingType.InsideRectangular gegeben. Dabei wird der nächste Punkt der Geometrietriangulation zu einem gegebenen Voxel berechnet und geprüft, ob sich dieser innerhalb des Voxels befindet. Auch FillingType.SquaredDistance findet den nächsten Punkt der Geometrietriangulation und prüft ob sich dieser innerhalb eines vorgegebenen Abstands zum Mittelpunkt des Voxels befindet. Diese Variante eignet sich für uniforme Voxel, liefert allerdings minderwertige Ergebnisse wenn die Voxelausdehnungen in die drei Raumrichtungen differieren. Die präzisesten Ergebnisse liefert der Enumerationswert FillingType.ByCollision, bei dem die interne Kollisionserkennung für den Voxelstatus aufgerufen wird. Nachteile können hier in der Effizienz beobachtet werden. Die Wahl des Berechnungstyps ist problemabhängig und kann an den jeweiligen Anwendungsfall angepasst werden.

Verbund aus achsenorientierten Hüllquadern



Die Klasse `BoundingBoxSet` dient zur Erstellung eines achsenorientierten Hüllquaderverbunds in drei Dimensionen. Übergabeparameter sind die Voxelisierung eines dreidimensionalen triangulierten Oberflächennetzes in Form eines `VoxelSet` Objektes und je nach ausgewähltem Verfahren eine Vereinigungsreihenfolge oder eine Quaderanzahl sowie wahlweise ein Optimierungsverfahren in Form einer Enumeration, welche die Werte OptimizationType.MILP oder OptimizationType.MIQP annehmen kann. Die Voxelisierung (siehe Abschnitt 4.2.2) bildet die Basis für die Erzeugung des Quaderverbundes, wobei dessen Qualität in erster Linie von deren Qualität abhängt. Die entwickelte Entwurfssprache bietet die folgenden Verfahren zur Erzeugung von Verbunden aus Hüllquadern.

Quaderverbund durch Paketierung Bei der Paketierung wird die Voxelisierung in Richtung einer Raumachse durchlaufen, wobei benachbarte Voxel zu Säulen zusammengefasst werden. Im nächsten Schritt werden diese in Richtung der zweiten Raumachse durchlaufen und bei Nachbarschaft und gleicher Länge zu Blöcken in Voxelhöhe zusammengefasst. Abschließend werden diese in Richtung der dritten und letzten Raumachse durchsucht und bei Nachbarschaft und gleicher Länge sowie Breite zu Quadern zusammengefasst. Input für die Methode

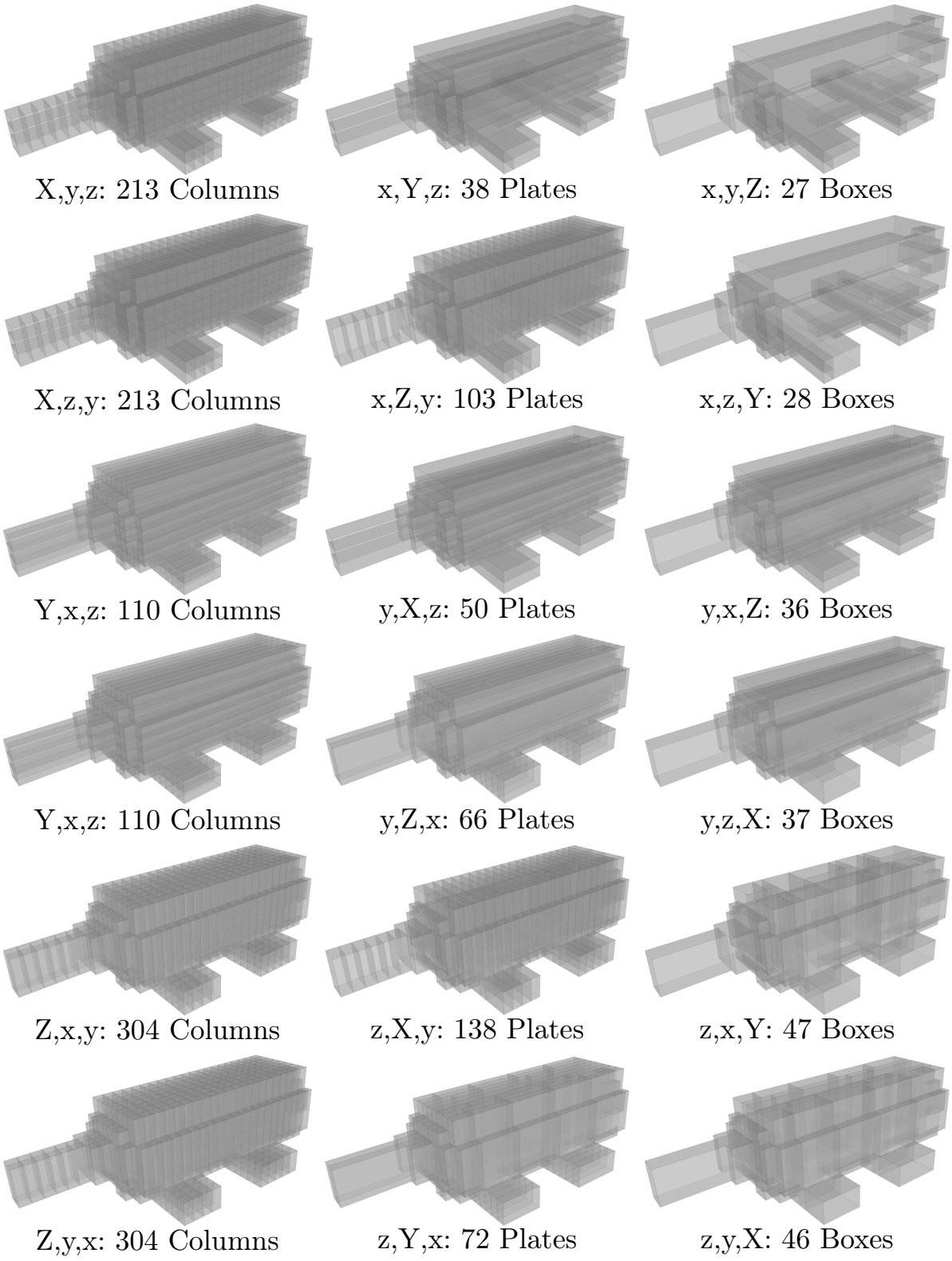


Abbildung 4.4: Quaderverbund eines Druckwandlers für Raumfahrtantriebe durch Paketierung einer 16x32x8 Voxelisierung entlang der Achsen in vorgegebener Reihenfolge. Von oben nach unten (Reihen v.l.n.r.) xyz, xzy, yxz, yzx, zxy und zyx.

des Zusammenfassens ist die Reihenfolge, in der die Raumachsen durchlaufen werden sollen. Die Bilderreihe Abb. 4.4 veranschaulicht die drei Schritte während der Erstellung des Quaderverbundes am Beispiel eines Druckwandlers für Raumfahrtantriebe mit einer für Darstellungszwecke grob gewählten Input-Voxelisierung von 16x32x8.

Optimierter Quaderverbund aus Voxelisierung Da die Implementierung einer Optimierungsmethode Bestandteil der vorliegenden Arbeit ist, wird diese auch zur Erstellung eines Quaderverbundes eingesetzt. Die entwickelte Methode ist an [Demiröz et al., 2019] angelehnt und wurde im Zuge der vorliegenden Arbeit nach 3D erweitert sowie an den Anwendungsfall angepasst. In [Demiröz et al., 2019] beschäftigen sich die Autoren mit dem sog. „Rectangle Blanket Problem“. Dieses beinhaltet die Ermittlung eines optimalen achsenorientierten Rechteckverbundes zur Vereinfachung einer zweidimensionalen Geometrie, wobei die sich nicht überlappenden Flächen zwischen Verbund und Geometrie minimiert werden sollen. Für die Berechnung wird die anzunähernde Geometrie zunächst pixelisiert und eine gemischt-ganzzahlige lineare Optimierungsmethode vorgestellt, in der die Pixel möglichst genau von dem zu optimierenden Rechteckverbund überdeckt sein sollen. Die gewünschte Anzahl der Rechtecke muss dabei vorgegeben werden. Der aus der Optimierung hervorgehende Verbund wird als eine sog. „Rechteckdecke“ bezeichnet und muss weder vollständig innerhalb der Geometrie enthalten sein, noch muss er die Geometrie vollständig bedecken. In der vorliegenden Arbeit ist die Formulierung des Optimierungsproblems insofern angepasst, dass dieses über die integrierte Schnittstelle übergeben und berechnet werden kann. Weiterhin wird für die Erweiterung auf ein 3D Problem statt der Pixelisierung eine Voxelisierung bestehend aus vollen und leeren Voxeln verwendet. Außerdem muss das 2D-Optimierungsproblem um zusätzliche Gleichungen für die dritte Raumdimension angereichert werden. Eine weitere Anpassung umfasst die Zusatzbedingung, dass alle vollen Voxel von dem zu berechnenden Quaderverbund umschlossen werden müssen. Dieser konservative Ansatz stellt bei anschließendem Einsatz einer Kollisionsüberprüfung sicher, dass Kollisionen zuverlässig erkannt werden und nicht durch die Vereinfachung der Geometrie in Fällen mit geringer Kollisionstiefe übersehen werden. Weiterhin dürfen sich die Quader im Gegensatz zu [Demiröz et al., 2019] gegenseitig verschneiden. Dadurch soll die Geometrieannäherung noch verbessert werden. Eine weitere Besonderheit ist die Vorverarbeitung der Voxelisierung. In einem Preprocessing Schritt werden alle inneren Voxel, welche nicht mit den Oberflächen des triangulierten Geometrienetzes schneiden herausgefiltert, da sie nicht relevant für die Optimierung des Quaderverbundes sind. Durch die Reduktion der in das Problem eingehenden Voxelanzahl sinkt die Anzahl der Gleichungen und somit die Laufzeit des Lösungsprozesses. Inspiriert von der Veröffentlichung [Demiröz et al., 2019] ergibt sich für den dreidimensionalen Fall also insgesamt folgendes gemischt-ganzzahliges lineares Optimierungsproblem für eine reduzierte Voxelisierung mit i Voxeln:

Indizes:

i: Index für das Voxel $i = 0, ..., I$
j: Index für das volle Voxel $j = 0, ..., J$
k: Index für das leere Voxel $k = J + 1, ..., J + K$
b: Index für die Box $b = 0, ..., B$

Parameter:

I: Anzahl aller Voxel
J: Anzahl der vollen Voxel
K: Anzahl der leeren Voxel, wobei $K = I - J$

B: Anzahl der verfügbaren Boxen
x_i^v, y_i^v, z_i^v: Position der linken vorderen unteren Ecke des Voxels i
L, W, H: Länge, Breite, Höhe der Voxelisierung
l^v, w^v, h^v: Länge, Breite, Höhe jedes Voxels

Kontinuierliche Variablen:

x_b, y_b, z_b: Position der linken vorderen unteren Ecke der Box b
l_b, w_b, h_b: Länge, Breite, Höhe der Box b.

Binäre Variablen für die relativen Positionen der Boxen zu den Voxeln:

$$X_{b,i}^- = \begin{cases} 1 & \text{die Box befindet sich links des Voxels,} \\ 0 & \text{Boxposition undefiniert} \end{cases} \tag{4.1}$$

$$X_{b,i}^+ = \begin{cases} 1 & \text{die Box befindet sich rechts des Voxels,} \\ 0 & \text{Boxposition undefiniert} \end{cases} \tag{4.2}$$

$$Y_{b,i}^- = \begin{cases} 1 & \text{die Box befindet sich vor dem Voxel,} \\ 0 & \text{Boxposition undefiniert} \end{cases} \tag{4.3}$$

$$Y_{b,i}^+ = \begin{cases} 1 & \text{die Box befindet sich hinter dem Voxel,} \\ 0 & \text{Boxposition undefiniert} \end{cases} \tag{4.4}$$

$$Z_{b,i}^- = \begin{cases} 1 & \text{die Box befindet sich unterhalb des Voxels,} \\ 0 & \text{Boxposition undefiniert} \end{cases} \tag{4.5}$$

$$Z_{b,i}^+ = \begin{cases} 1 & \text{die Box befindet sich oberhalb des Voxels,} \\ 0 & \text{Boxposition undefiniert} \end{cases} \tag{4.6}$$

$$V_{b,i} = \begin{cases} 1 & \text{die Box überdeckt das Voxel,} \\ 0 & \text{Boxposition undefiniert} \end{cases} \tag{4.7}$$

Zielfunktion:

$$min \sum_{b=0}^{B} \sum_{k=J+1}^{J+K} V_{b,k} \tag{4.8}$$

Gleichungen:

$$x_b + l_b \leq x_i^v + (1 - X_{b,i}^-) \cdot L \qquad b = 0, ..., B;\ i = 0, ..., I \tag{4.9}$$

$$x_i^v + l^v \leq x_b + (1 - X_{b,i}^+) \cdot L \qquad b = 0, ..., B;\ i = 0, ..., I \tag{4.10}$$

$$y_b + w_b \leq y_i^v + (1 - Y_{b,i}^-) \cdot W \qquad b = 0, ..., B;\ i = 0, ..., I \tag{4.11}$$

$$y_i^v + w^v \leq y_b + (1 - Y_{b,i}^+) \cdot W \qquad b = 0, ..., B;\ i = 0, ..., I \tag{4.12}$$

$$z_b + h_b \leq z_i^v + (1 - Z_{b,i}^-) \cdot H \qquad b = 0, ..., B;\ i = 0, ..., I \tag{4.13}$$

$$z_i^v + h^v \leq z_b + (1 - Z_{b,i}^+) \cdot H \qquad b = 0, ..., B;\ i = 0, ..., I \tag{4.14}$$

$$x_b \geq x_i^v + (1 - V_{b,i}) \cdot L \qquad b = 0, ..., B;\ i = 0, ..., I \tag{4.15}$$

$$x_i^v + l^v \geq x_b + l_b + (1 - V_{b,i}) \cdot L \qquad b = 0, ..., B;\ i = 0, ..., I \tag{4.16}$$

$$y_b \geq y_i^v + (1 - V_{b,i}) \cdot W \qquad b = 0, ..., B;\ i = 0, ..., I \tag{4.17}$$

$$y_i^v + w^v \geq y_b + w_b + (1 - V_{b,i}) \cdot W \qquad b = 0, ..., B;\ i = 0, ..., I \tag{4.18}$$

$$z_b \geq z_i^v + (1 - V_{b,i}) \cdot H \qquad b = 0, ..., B;\ i = 0, ..., I \tag{4.19}$$

$$z_i^v + h^v \geq z_b + h_b + (1 - V_{b,i}) \cdot H \qquad b = 0, ..., B;\ i = 0, ..., I \tag{4.20}$$

$$X_{b,i}^- + X_{b,i}^+ + Y_{b,i}^- + Y_{b,i}^+ + Z_{b,i}^- + Z_{b,i}^+ + V_{b,i} = 1 \qquad b = 0, ..., B;\ i = 0, ..., I \tag{4.21}$$

$$\sum_{b=0}^{B} V_{b,j} \geq 1 \qquad j = 0, ..., J \tag{4.22}$$

$$x_b \leq L - l^v \qquad b = 0, ..., B \tag{4.23}$$

$$y_b \leq W - w^v \qquad b = 0, ..., B \tag{4.24}$$

$$z_b \leq H - h^v \qquad b = 0, ..., B \tag{4.25}$$

$$x_b + l_b \leq L \qquad b = 0, ..., B \tag{4.26}$$

$$y_b + w_b \leq W \qquad b = 0, ..., B \tag{4.27}$$

$$z_b + h_b \leq H \qquad b = 0, ..., B \tag{4.28}$$

In der vorgestellten Problemformulierung beschreibt Gl. (4.8) die Zielfunktion, in der die Anzahl der im Quaderverbund enthaltenen leeren Voxel minimiert werden soll. Aus Gleichungen (4.9) bis (4.14) greift genau eine, wenn die Box b das Voxel i nicht überdeckt. Gleichungen (4.15) bis (4.20) sind dagegen alle gleichzeitig aktiv, wenn die Box b das Voxel i beinhaltet. Gl. (4.21) stellt sicher, dass jede Box relativ zu jedem Voxel mindestens einen der aufgeführten Zustände einnimmt. Dabei schließt der Wert 0 einen Zustand nicht zwingend aus, so kann sich eine Box bei $Z_{b,i}^+ = 1$ zwar nicht gleichzeitig oberhalb und unterhalb eines Voxels befinden (den Zustand „unterhalb“ schließt Gl. (4.21) aus), sie kann allerdings durchaus links-oberhalb oder rechts-oberhalb des Voxels liegen. Laut Gl. (4.22) muss jedes volle Voxel innerhalb mindestens eines Quaders enthalten sein. Gleichungen (4.23) bis (4.28) beschreiben die unteren und oberen Grenzen der einzelnen Variablen des Gesamtproblems. Zur Lösung des Problems kann eine beliebige Optimierungsmethode aus den in der vorliegenden Arbeit entwickelten Optionen ausgewählt werden. In der Standardeinstellung wird ein Open-Source-Solver verwendet.

Optimierter Quaderverbund aus Teilvoxelisierung Die Optimierung unter Berücksichtigung einer Voxelisierung ist durch potenziell große Voxelanzahlen extrem rechenaufwändig. Für eine Verbesserung der Rechenzeit muss die Anzahl der Voxel und dadurch die Anzahl der Gleichungen reduziert werden, bestenfalls ohne eine Verschlechterung der Voxelisierungsqualität. In diesem Kontext ist eine weitere Methode implementiert, welche ausschließlich die für die Geometrieapproximation relevanten Voxel berücksichtigt. Dabei werden neben den inneren und

damit nichtrelevanten Voxel, auch die außerhalb der Geometrie liegenden Voxel vernachlässigt. Gehen letztere nicht in das Problem ein, muss eine neue Zielfunktion entwickelt werden. Diese soll nun das Gesamtvolumen des Quaderverbundes minimieren, um eine möglichst enge Umhüllende der Geometrie zu gewährleisten. Ein Nachteil der vorgestellten Vorgehensweise ist die Nichtlinearität des Gesamtproblems durch die Volumenkalkulation. Durch Einsatz von Reformulierung (siehe 4.3.3) kann das kubische Gesamtproblem allerdings um einen Grad reduziert und in ein quadratisches Problem überführt werden. Welche Methode die geeignetere ist, muss für den Einzelfall entschieden werden, insgesamt überwiegt jedoch in vielen Anwendungsfällen der Vorteil durch die Reduktion der Gleichungsanzahl den Nachteil durch den höheren Grad des Gesamtproblems. Es ergeben sich die folgenden Änderungen, welche zu einem gemischt-ganzzahligen quadratischen Optimierungsproblem mit linearen Anteilen führen:

- Es wird die Ersatzvariable A_b für die Fläche der Box b eingeführt. Diese wird benötigt, um die Volumenberechnung in der Zielfunktion als quadratische Gleichung auszudrücken.
- Es wird eine neue Zielfunktion zur Minimierung des Verbundvolumens eingeführt:

$$min \sum_{b=0}^{B} A_b \cdot h_b \tag{4.29}$$

- Es wird eine zusätzliche Gleichung benötigt, welche die Ersatzvariable A_b als Produkt aus Fläche und Breite der Box b darstellt.

$$w_b \cdot l_b - A_b = 0 \qquad b = 0, ..., B \tag{4.30}$$

- Gl. (4.22) gilt nun für alle in das Gesamtproblem eingehenden Voxel.

$$\sum_{b=0}^{B} V_{b,i} \geq 1 \qquad i = 0, ..., I \tag{4.31}$$

- Gl. (4.9) bis Gl. (4.21) und Gl. (4.23) bis Gl. (4.28) bleiben erhalten.

Optimierter Quaderverbund aus Paketierung Für eine weitere Steigerung der Effizienz ist eine Methode implementiert, welche sich in einem Vorverarbeitungsschritt die bereits vorgestellte Paketierung aus Abschnitt 4.2.2 zu Nutzen macht. Statt der Voxel sollen nun alle daraus gewonnenen Pakete in das Optimierungsproblem eingehen. Dadurch kann die Anzahl an Gleichungen und damit die Rechenzeit enorm reduziert werden. Es ergibt sich nun folgendes gemischt-ganzzahliges linear-quadratisches Optimierungsproblem für die Paketierung:

Indizes:

i: Index für das Paket $i = 0, ..., I$
b: Index für die Box $b = 0, ..., B$

Parameter:

I: Anzahl der Pakete
B: Anzahl der verfügbaren Boxen
x_i^p, y_i^p, z_i^p: Position der linken vorderen unteren Ecke des Pakets i

L, W, H: Länge, Breite, Höhe der Paketierung
l^p, w^p, h^p: Länge, Breite, Höhe jedes Pakets

Kontinuierliche Variablen:

x_b, y_b, z_b: Position der linken vorderen unteren Ecke der Box b
l_b, w_b, h_b: Länge, Breite, Höhe der Box b
A_b: Ersatzvariable für die Fläche aus Länge · Breite der Box b

Binäre Variablen für die relativen Positionen der Boxen zu den Paketen:

$$X^-_{b,i} = \begin{cases} 1 & \text{die Box befindet sich links des Pakets,} \\ 0 & \text{Boxposition undefiniert} \end{cases} \tag{4.32}$$

$$X^+_{b,i} = \begin{cases} 1 & \text{die Box befindet sich rechts des Pakets,} \\ 0 & \text{Boxposition undefiniert} \end{cases} \tag{4.33}$$

$$Y^-_{b,i} = \begin{cases} 1 & \text{die Box befindet sich vor dem Paket,} \\ 0 & \text{Boxposition undefiniert} \end{cases} \tag{4.34}$$

$$Y^+_{b,i} = \begin{cases} 1 & \text{die Box befindet sich hinter dem Paket,} \\ 0 & \text{Boxposition undefiniert} \end{cases} \tag{4.35}$$

$$Z^-_{b,i} = \begin{cases} 1 & \text{die Box befindet sich unterhalb des Pakets,} \\ 0 & \text{Boxposition undefiniert} \end{cases} \tag{4.36}$$

$$Z^+_{b,i} = \begin{cases} 1 & \text{die Box befindet sich oberhalb des Pakets,} \\ 0 & \text{Boxposition undefiniert} \end{cases} \tag{4.37}$$

$$V_{b,i} = \begin{cases} 1 & \text{die Box überdeckt das Paket,} \\ 0 & \text{Boxposition undefiniert} \end{cases} \tag{4.38}$$

Zielfunktion:

$$min \sum_{b=0}^{B} A_b \cdot h_b \tag{4.39}$$

Gleichungen:

$$x_b + l_b \leq x^p_i + (1 - X^-_{b,i}) \cdot L \qquad b = 0, ..., B;\ i = 0, ..., I \tag{4.40}$$

$$x^p_i + l^p \leq x_b + (1 - X^+_{b,i}) \cdot L \qquad b = 0, ..., B;\ i = 0, ..., I \tag{4.41}$$

$$y_b + w_b \leq y^p_i + (1 - Y^-_{b,i}) \cdot W \qquad b = 0, ..., B;\ i = 0, ..., I \tag{4.42}$$

$$y^p_i + w^p \leq y_b + (1 - Y^+_{b,i}) \cdot W \qquad b = 0, ..., B;\ i = 0, ..., I \tag{4.43}$$

$$z_b + h_b \leq z^p_i + (1 - Z^-_{b,i}) \cdot H \qquad b = 0, ..., B;\ i = 0, ..., I \tag{4.44}$$

$$z^p_i + h^p \leq z_b + (1 - Z^+_{b,i}) \cdot H \qquad b = 0, ..., B;\ i = 0, ..., I \tag{4.45}$$

$$x_b \geq x_i^p + (1 - V_{b,i}) \cdot L \qquad b = 0, ..., B;\ i = 0, ..., I \tag{4.46}$$

$$x_i^p + l^p \geq x_b + l_b + (1 - V_{b,i}) \cdot L \qquad b = 0, ..., B;\ i = 0, ..., I \tag{4.47}$$

$$y_b \geq y_i^p + (1 - V_{b,i}) \cdot W \qquad b = 0, ..., B;\ i = 0, ..., I \tag{4.48}$$

$$y_i^p + w^p \geq y_b + w_b + (1 - V_{b,i}) \cdot W \qquad b = 0, ..., B;\ i = 0, ..., I \tag{4.49}$$

$$z_b \geq z_i^p + (1 - V_{b,i}) \cdot H \qquad b = 0, ..., B;\ i = 0, ..., I \tag{4.50}$$

$$z_i^p + h^p \geq z_b + h_b + (1 - V_{b,i}) \cdot H \qquad b = 0, ..., B;\ i = 0, ..., I \tag{4.51}$$

$$X_{b,i}^- + X_{b,i}^+ + Y_{b,i}^- + Y_{b,i}^+ + Z_{b,i}^- + Z_{b,i}^+ + V_{b,i} = 1 \qquad b = 0, ..., B;\ i = 0, ..., I \tag{4.52}$$

$$\sum_{b=0}^{B} V_{b,i} \geq 1 \qquad i = 0, ..., I \tag{4.53}$$

$$w_b \cdot l_b - A_b = 0 \qquad b = 0, ..., B \tag{4.54}$$

$$x_b \leq L - l^p \qquad b = 0, ..., B \tag{4.55}$$

$$y_b \leq W - w^p \qquad b = 0, ..., B \tag{4.56}$$

$$z_b \leq H - h^p \qquad b = 0, ..., B \tag{4.57}$$

$$x_b + l_b \leq L \qquad b = 0, ..., B \tag{4.58}$$

$$y_b + w_b \leq W \qquad b = 0, ..., B \tag{4.59}$$

$$z_b + h_b \leq H \qquad b = 0, ..., B \tag{4.60}$$

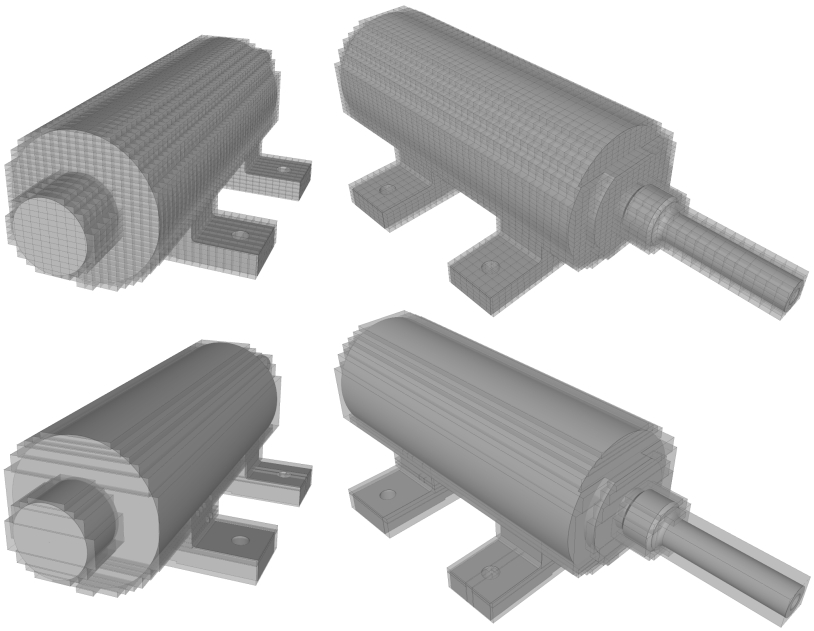


Abbildung 4.5: Vorverarbeitungsschritte bei der Berechnung eines optimierten Quaderverbundes aus Paketierung für einen Druckwandler für Raumfahrtantriebe. Oben: Voxelisierung 32x64x32 (21373 Voxel). Unten: Paketierung (71 Pakete).

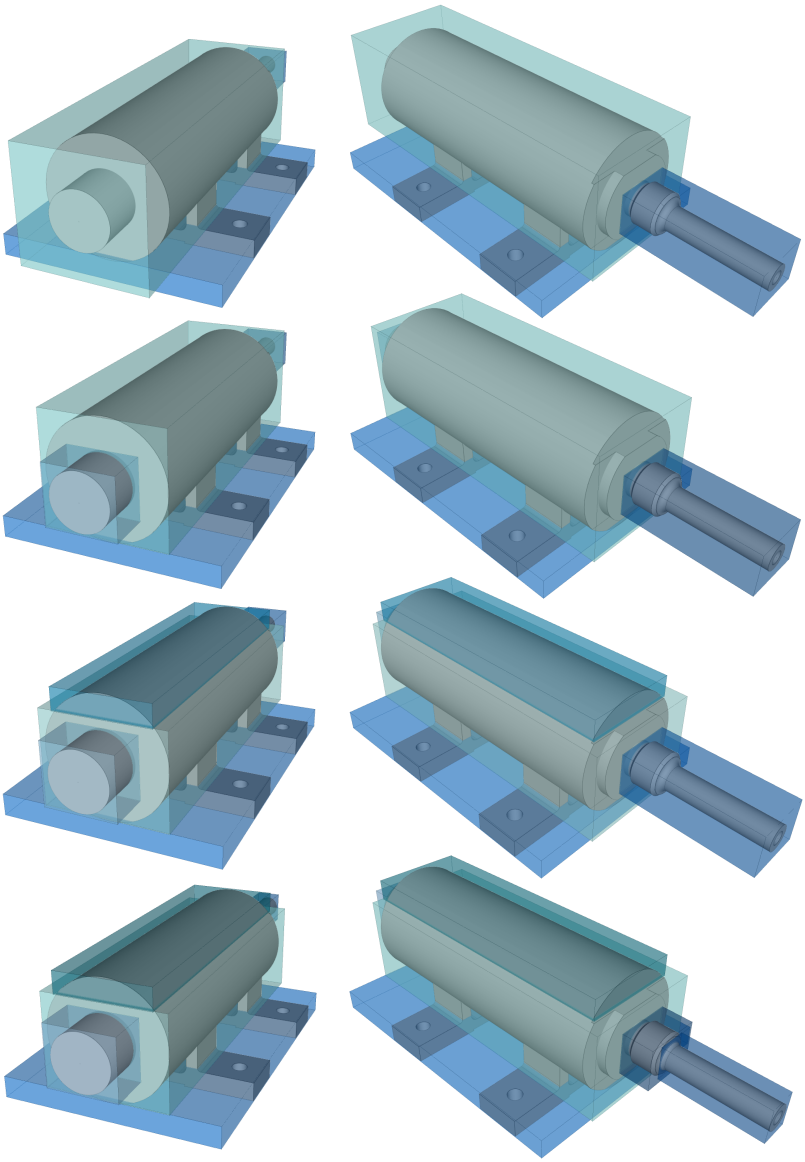


Abbildung 4.6:
Varianten des optimierten Quaderverbundes aus Paketierung als geometrische Annäherung eines Druckwandlers für Raumfahrtantriebe.
V.o.n.u.:
3 optimale Quader,
4 optimale Quader,
5 optimale Quader
und 6 optimale Quader.

In der vorgestellten Problemformulierung beschreibt Gl. (4.39) die Zielfunktion, in der das Gesamtvolumen des Quaderverbundes minimiert werden soll. Aus Gleichungen (4.40) bis (4.45) greift genau eine, wenn die Box b das Paket i nicht überdeckt. Gleichungen (4.46) bis (4.51) sind dagegen alle gleichzeitig aktiv, wenn die Box b das Paket i beinhaltet. Gl. (4.52) stellt sicher, dass jede Box relativ zu jedem Paket mindestens einen der aufgeführten möglichen Zustände einnimmt. Laut Gl. (4.53) muss jedes Paket innerhalb eines Quaders des Quaderverbundes enthalten sein. Gl. (4.54) wird für die Flächenberechnung benötigt, um die Volumenberechnung in der Zielfunktion als quadratische Gleichung ausdrücken zu können. Gleichungen (4.55) bis (4.60) beschreiben die unteren und oberen Grenzen der einzelnen Variablen des Gesamtproblems. Wie in der zuvor beschriebenen Quaderverbundoptimierung kann zur Lösung des Problems eine beliebige Optimierungsmethode aus den in der vorliegenden Arbeit entwickelten Optionen ausgewählt werden. Die Bilderreihe in Abschnitt 4.2.2 und Abb. 4.6 stellt die Schritte für die Optimierung des Quaderverbundes dar und zeigt berechnete Lösungen mit ansteigender Quaderanzahl am Beispiel eines Druckwandlers für Raumfahrtantriebe.

Beliebig orientierter Hüllquader

MinimumVolumeBoundingBox
corner : double[ ] minEdge : double[ ] midEdge : double[ ] maxEdge : double[ ]
MinimumVolumeBoundingBox(input : GAMesh)

Die Klasse `MinimumVolumeBoundingBox` dient zur Erstellung eines beliebig orientierten Hüllquaders mit minimalem Volumen zur möglichst präzisen Annäherung einer dreidimensionalen Geometrie. Zur Erstellung muss lediglich das dreidimensionale Oberflächennetz in Form eines `GAMesh` Objektes an den Konstruktor übergeben werden. Für die Berechnung des Hüllquaders werden alle im Oberflächennetz enthaltenen Eckpunkte an ein `vtkOBBTree` Objekt aus der *VTK* Bibliothek [Schroeder et al., 2006] übergeben. Die `vtkOBBTree` *VTK* Klasse implementiert einen Algorithmus zur Bestimmung der Kovarianzmatrix der Punktewolke und derer Eigenwerte, welche die drei Richtungsvektoren der maximalen Ausdehnung der Punkte beschreiben ([Schroeder et al., 2006]). Durch anschließende Projektion der Punkte kann daraus der am engsten passende und somit bezüglich des Volumens minimale orientierte Hüllquader berechnet werden.

Konvexe Hülle

ConvexHull
hull : GAMesh
ConvexHull(input : GAMesh) ConvexHull(input : GAMesh, scale : double)

Die Klasse `ConvexHull` dient zur Erstellung einer konvexen Hülle in drei Dimensionen. Übergabeparameter sind ein dreidimensionales Oberflächennetz in Form eines `GAMesh` Objektes und ein optionaler Skalierungsparameter. Die Basis für die Erstellung der konvexen Hülle bildet die Klasse `ConvexHullSet` (siehe *4.2.2 Verbund aus konvexen Hüllen*), wobei bei Aufruf des Konstruktors der Klasse `ConvexHullSet` durch den `ConvexHull` Konstruktor die gewünschte Hüllenanzahl gleich 1 als Parameter übergeben wird, sodass genau eine konvexe Hülle berechnet wird. Bei Übergabe eines Skalierungsparameters wird auch hier der Algorithmus zur Nachbearbeitung der Hülle zugeschaltet.

Verbund aus konvexen Hüllen

ConvexHullSet
hulls : GAMesh[ ]
ConvexHullSet(input : GAMesh) ConvexHullSet(input : GAMesh, hullNumber : int) ConvexHullSet(input : GAMesh, hullNumber : int, scale : double[ ])

Die Klasse `ConvexHullSet` dient zur Erzeugung eines Verbundes aus konvexen Hüllen. Übergabeparameter sind ein dreidimensionales Oberflächennetz in Form einer `GAMesh` Objektes, eine optionale Hüllenanzahl und ein optionaler Skalierungsparameter. Für die Berechnung der konvexen Hüllen wird zunächst der *volumetric hierarchical approximate convex decomposition* Algorithmus aus der frei verfügbaren Bibliothek *V-HACD* [Mamou und Ghorbel, 2009] verwendet. Da exakte konvexe Zerlegungsalgorithmen im allgemeinen NP-schwer sind, sind sie für viele Anwendungsfälle unpraktikabel ([Mamou und Ghorbel, 2009]). In der *V-HACD* Bibliothek wird daher die exakte Konvexitätsbeschränkung gelockert und stattdessen eine nur annähernd konvexe Zerlegung des eingehenden Oberflächennetzes berechnet ([Mamou und Ghorbel, 2009]). Ziel ist die Aufteilung der Triangulierung durch eine minimale Anzahl von Polyedern, wobei gleichzeitig sichergestellt

wird, dass jeder Polyeder eine maximale Nichtkonvexität aufweist, die unter einem vom Benutzer festgelegten Schwellenwert liegt ([Mamou und Ghorbel, 2009]). Die Stärke der verwendeten Bibliothek liegt in ihrer Fähigkeit, vorab die maximal gewünschte Anzahl der Polyederecken einzuschränken. Dadurch kann die Komplexität der Ergebnisse an den jeweiligen Anwendungsfall angepasst werden. Die Erstellung mit Hilfe des genannten Algorithmus bietet also enorme Vorteile hinsichtlich der Effizienz, kann jedoch wie beschrieben nicht garantieren, dass die berechneten Hüllen vollständig konvex sind. Um Konvexität zu erreichen, werden die erzeugten Polyeder in einem nachfolgenden Schritt erneut konvex vernetzt. In den meisten Anwendungsfällen der vorliegenden Arbeit liegen die so erzeugten konvexen Polyeder allerdings in einigen Arealen innerhalb der Geometrie oder es treten Spalte zwischen ihnen auf. Um das vollständige Enthaltensein der Originalgeometrie innerhalb des Hüllenverbundes und die Spaltfreiheit zwischen den Hüllen zu gewährleisten, ist ein Algorithmus zur Nachbearbeitung der Hüllen implementiert, welcher bei Übergabe eines Skalierungsparameters zugeschaltet wird. Dieser Algorithmus skaliert die erzeugten konvexen Polyeder um ihre eigenen Mittelpunkte, sodass alle Anteile der Geometrietriangulation in dem Hüllenverbund enthalten sind. Im besten Fall errechnet sich der Wert des Skalierungsparameters aus näherungsweise der Hälfte der Spalte zwischen den Hüllen. Die Bilderreihe in Abb. 4.7 zeigt die erzielten Ergebnisse am Beispiel eines Antriebs für Vorflügelklappen und eines pyrotechnischen Ventils für Raumfahrtantriebe.

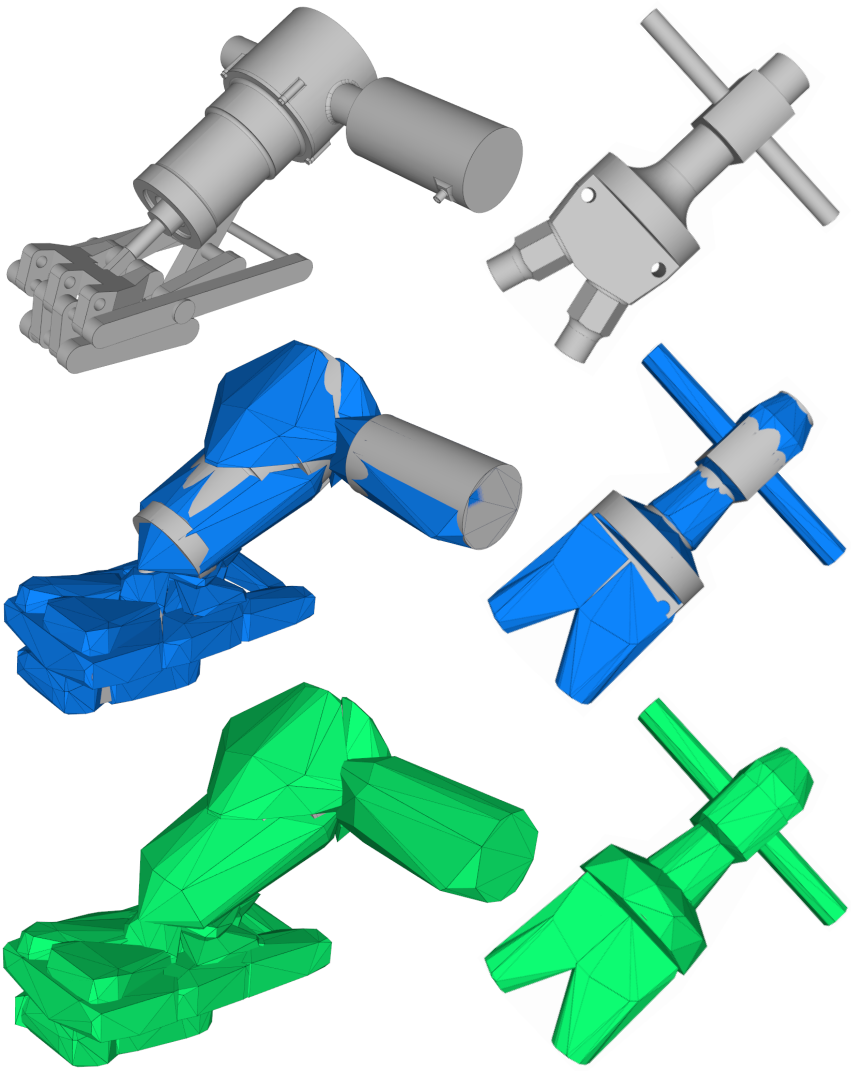
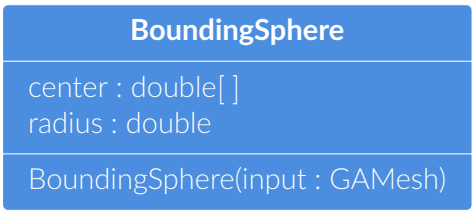


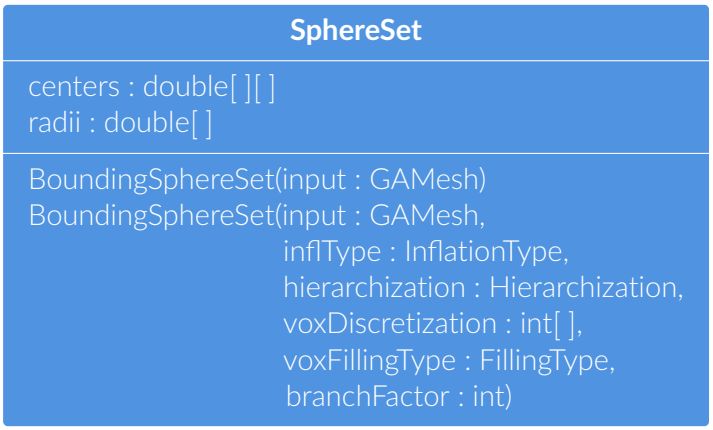
Abbildung 4.7: Verbund aus konvexen Hüllen. Linke Spalte: Antrieb für Vorflügelklappen. Rechte Spalte: Pyrotechnisches Ventil für Raumfahrtantriebe. V.o.n.u: Originalgeometrie, Ergebnis nach Erzeugung der konvexen Hüllen, Ergebnis nach Anwendung der zusätzlichen Skalierung.

Hüllkugel



Die Klasse `BoundingSphere` dient zur Erstellung einer umhüllenden Kugel eines dreidimensionalen triangulierten Oberflächennetzes. Für die Berechnung der Hüllkugel werden alle im Oberflächennetz enthaltenen Eckpunkte an die Klasse `WelzlEncloser` aus der frei verfügbaren Bibliothek *Apache Commons* [The Apache Software Foundation, 2022] übergeben. Die Klasse implementiert den Algorithmus aus [Welzl, 1991] inklusive der Modifikationen aus [Gärtner, 1999] und [Larsson und Källberg, 2013].

Verbund aus Hüllkugeln



Die Klasse `SphereSet` dient zur Erstellung eines Verbundes aus Kugeln, welche eine Geometrie approximieren. Je nach Anforderungen kann nicht nur der Verbund selbst sondern dessen hierarchische Baumstruktur nützlich sein, der sog. „Sphere-Tree“. Wird in der Vorverarbeitung ein Kugelverbundbaum aufgebaut, kann zur Laufzeit entschieden werden, ob der gesamte Baum für die Kollisionserkennung verwendet wird oder nur der Hüllkugelverbund auf einer der Ebenen. Die in der vorliegenden Arbeit entwickelten Algorithmen zur Erstellung des Baumes erweitern die Methoden von [Weller und Zachmann, 2008]. Dort wird eine Methode vorgestellt, in welcher die anzunähernde Geometrie zunächst voxelisiert wird, um daraus ein Set an Kugeln zu gewinnen. Diese werden anschließend mit einem vorgegebenen Verzweigungsfaktor ebenenweise baumartig zu weiteren Hüllkugeln zusammengefasst, bis eine hierarchische Struktur entsteht. Dabei ist zu beachten, dass eine Kugel auf der obersten Ebene nicht die zugehörigen Kugeln eine eben tiefer umschließen muss, sondern alle deren Nachfahren, welche sich auf der untersten Ebene befinden. Diese in [Weller und Zachmann, 2008] vorgestellten Kugelstrukturen werden als sog. „Inner-Sphere-Trees“ bezeichnet. Der Kugelverbund auf der untersten Ebene zeichnet sich dadurch aus, dass er vollständig in der zu approximierenden Geometrie enthalten ist. Für eine konservative Kollisionserkennung entsteht der Nachteil, dass Kollisionen von Geometrien potenziell unentdeckt bleiben, wenn Areale kollidieren, die nicht durch eine Kugel ausgefüllt sind. Die in der vorliegenden Arbeit entwickelten Erweiterungen modifizieren den mittels der Algorithmen aus [Weller und Zachmann, 2008] berechneten „Inner-Sphere-Tree“ insoweit, dass die gesamte zu approximierende Geometrie vollständig darin enthalten ist. Die vorgestellte Methode wendet dafür ein Aufblähen der Kugeln an, wodurch ein sog. „Inflated-Sphere-Tree“ entsteht. Das Ausdehnen bzw. Aufblähen der Kugeln findet mit einem von zwei zur Verfügung stehenden im Laufe der vorliegenden Arbeit entwickelten Algorithmen statt, welcher über die Enumeration InflationType als Input vorgegeben wird. Für die Erstellung der hierarchischen Baumstruktur sind ebenfalls zwei Verfahren implementiert. Diese können durch eine Übergabe der Enumeration Hierarchization an den Konstruktor der `SphereSet` Klasse spezifiziert werden. Weitere Übergabeparameter sind die Diskretisierung der Voxelisierung und optional der Verzweigungsfaktor, welcher defaultmäßig gleich 4 gesetzt ist. Für die Voxelisierung kann optional aus einer Bandbreite an Berechnungsarten für die Voxelfüllung bzw. den Voxelstatus ausgewählt werden (vgl. 4.2.2).

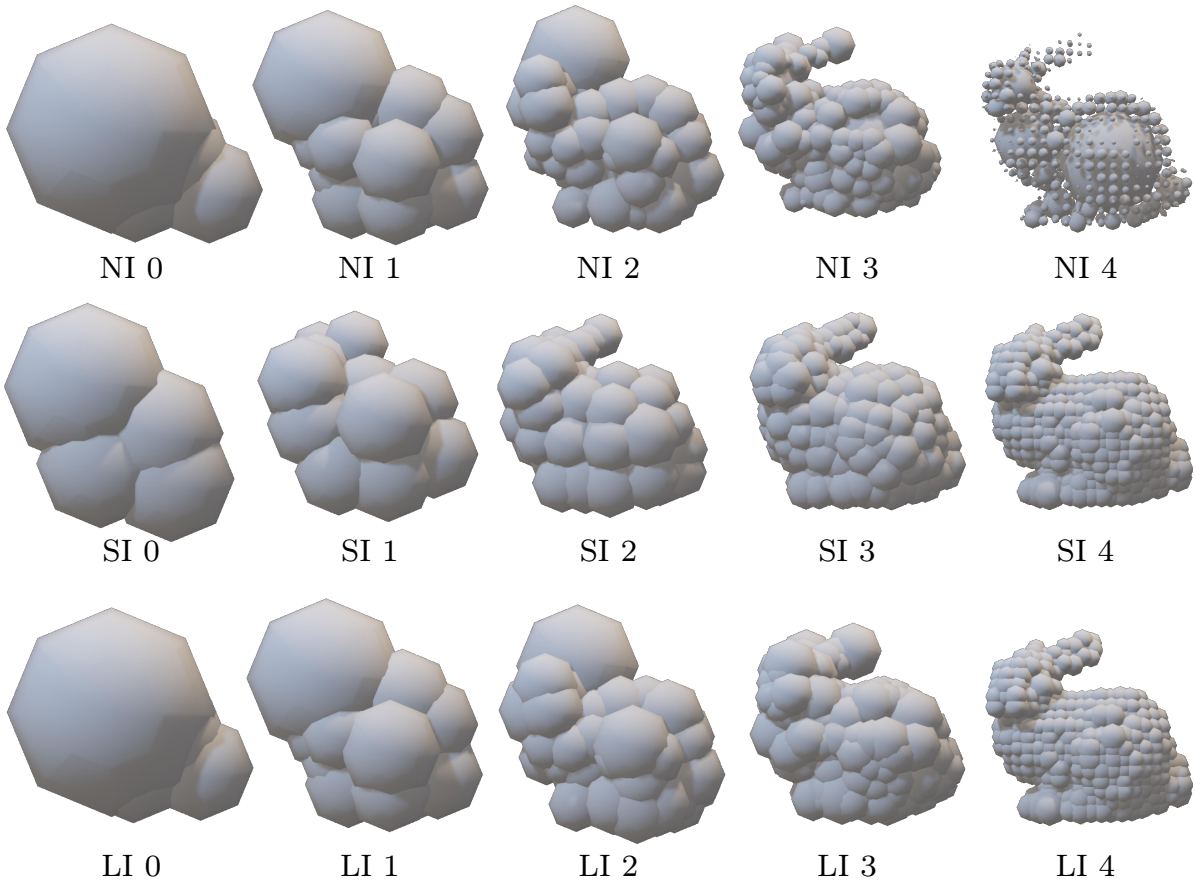


Abbildung 4.8: Modifizierte Sphere-Tree-Erstellung. Von oben nach unten die Varianten ohne Aufblähen der Kugeln (NI - NoInflation) nach [Weller und Zachmann, 2008], mit Aufblähen der untersten Ebene und anschließender Hierarchisierung (SI - StartInflation) und separatem Aufblähen auf jeder Ebene (LI - LevelInflation). Angabe der jeweiligen Ebene hinter der Variantenbezeichnung.

Ausdehnung durch Ausnutzung der Triangulation Wird der Enumerationswert InflationType.Triangle an den Konstruktor der `SphereSet` Klasse übergeben, werden die Dreiecke des Oberflächennetzes zur Skalierung der Kugeln eingesetzt. Jedem Dreieck wird dazu die am nächsten liegende Kugel zugeordnet. Im Anschluss werden alle Kugeln so weit skaliert, bis alle zugehörigen Dreiecke von mindestens einer der Kugeln umschlossen sind. Sind Kugeln in anderen Kugeln enthalten, werden sie gelöscht. Im Fall von langgestreckten Geometrien muss die Triangulation davor verfeinert werden, da extrem lange Dreiecke zu unverhältnismäßig großen Skalierungen der Kugeln führen.

Ausdehnung über Konservative Voxelisierung Wird der Enumerationswert InflationType.Voxel an den Konstruktor der `SphereSet` Klasse übergeben, werden die Kugeln im Anschluss an ihre Erzeugung nach [Weller und Zachmann, 2008] so weit skaliert, bis alle Voxel komplett darin enthalten sind. Für die Kugelausdehnung wird eine konservative Voxelisierungsvariante gewählt. In dieser werden sowohl alle Voxel befüllt, deren Mittelpunkte innerhalb der Geometrie enthalten sind, als auch die, deren Oberflächen die Oberfläche der Geometrie schneiden. Ist für die ursprüngliche Voxelisierung der Enumerationswert FillingType.Center als Input gewählt, wird hierfür automatisch auf eine der konservativen Füllungsarten umgestellt.

Hierarchisierung Wird Hierarchization.STARTINFLATION an den Konstruktor der `SphereSet`-Klasse übergeben, wird der Hierarchieaufbau nach [Weller und Zachmann, 2008] mit einem aufgeblähten Kugelverbund auf unterster Ebene ausgeführt, was sich auf das Ergebnis in jeder weiteren Hierarchieebene auswirkt. Wird Hierarchization.LEVELINFLATION übergeben, wird die Hierarchie nach [Weller und Zachmann, 2008] aufgebaut und erst im Anschluss jede Ebene separat und unabhängig von den anderen Ebenen aufgebläht. In Abb. 4.8 wird die Hierarchisierung nach [Weller und Zachmann, 2008] den beiden entwickelten Methoden am Beispiel des Stanford Bunny gegenübergestellt. Soll keine Modifikation des originalen Algorithmus statt finden, muss der Enumerationswert Hierarchization.NOINFLATION übergeben werden. Diese Variante ist in der ersten Zeile von Abb. 4.8 dargestellt.

Zweidimensionale Geometrien und Reduktion der Dimension

Im Folgenden werden verschiedene Geometrievereinfachungen in 2D vorgestellt. Als Input können neben zweidimensionalen auch dreidimensionale triangulierte Oberflächennetze verarbeitet werden. Im Falle eines dreidimensionalen Netzes muss zur Spezifikation der zu vernachlässigenden Dimension eine sog. Projektionsebene in Form einer Enumeration übergeben werden. Dabei stehen die Ebenen ProjectonPlane.XY, ProjectionPlane.XZ und ProjectionPlane.YZ zur Verfügung. Neben der Projektionsebene kann ein Verschiebungsparameter übergeben, welcher die Verschiebung der Ebene im Raum in Richtung ihrer Normalen angibt. Grundsätzlich stehen zwei Verfahren zur Verfügung, um die Dimension zu reduzieren. Die Auswahl erfolgt mithilfe eines sog. Dimensionsreduktionstyps als Enumeration. Dabei stehen die Typen DimensionReduction.PROJECTED und DimensionReduction.SLICED zur Verfügung.

3D-Triangulation zu 2D-Projektion Bei Übergabe eines dreidimensionalen Oberflächennetzes und des Parameters DimensionReduction.PROJECTED wird unter Berücksichtigung der gewünschten Projektionsebene die zweidimensionale Geometrieapproximation näherungsweise um die Projektion des 3D Netzes in dieser Ebene gebildet. Diese Funktionalität zielt auf eine Reduktion der räumlichen Dimension des Packingproblems von 3D nach 2D ab. Das Umhüllen der Projektion stellt sicher, dass die größten Ausmaße des dreidimensionalen Objektes in die niedrigere Dimensionalität übernommen werden. Ein mögliches Anwendungsszenario ist der Einsatz bei einem zweistufigen Layout-Optimierungsverfahren. Dieses kann aus einer zeitsparenden vorläufigen zweidimensionale Auslegung bestehen, bei der die Positionierung der Objekte in zwei Dimensionen konservativ mit den maximalen Körperausmaßen stattfindet, und einem nachgelagerten Schritt in drei Dimensionen, bei dem für eine genauere Nachjustierung der Positionen eine lokale Optimierung zum Einsatz kommen kann. Dadurch ließe sich eine aufwendige und rechenzeitintensive Optimierung ausschließlich im dreidimensionalen Raum vermeiden. Ein konkretes Anwendungsbeispiel für die zweidimensionale Geometrievereinfachung über die Projektion einer dreidimensionalen Oberfläche folgt in Abschnitt 4.2.2.

3D-Triangulation zu 2D-Querschnitt Wird neben der dreidimensionalen Oberflächentriangulation der Parameter DimensionReduction.SLICED übergeben, wird die zweidimensionale Geometrieapproximation näherungsweise um den Querschnitt des 3D-Netzes in dieser verschobenen Ebene erstellt. Auch diese Funktionalität dient der Reduktion der räumlichen Dimension zur Vereinfachung des Packagingproblems. Ein möglicher Anwendungsfall ist ein Szenario, in dem eine von drei Raumkoordinaten bei der Optimierung eine stark untergeordnete Rolle spielt oder bereits bekannt ist. Stehen zur Platzierung benötigte Hindernisse nur als dreidimensionale Dateien zur Verfügung, können zweidimensionale Schnitte von diesen auf Höhe der bekannten Koordinate berechnet und für eine zweidimensionale Platzierung verwendet werden.

Achsenorientiertes Hüllrechteck

AxisAlignedBoundingRectangle
minCoordiantes : double[ ] maxCoordiantes : double[ ]
AxisAlignedBoundingRectangle(input : GAPlanarMesh) AxisAlignedBoundingRectangle(input : GAMesh, plane : ProjectionPlane, dim : DimensionReduction, shift : double)

Die Klasse `AxisAlignedBoundingRectangle` dient zur Erstellung eines achsenorientierten Hüllrechtecks in zwei Dimensionen. Übergabeparameter sind ein zweidimensionale oder dreidimensionale Oberflächentriangulation in Form eines `GAMesh2D` oder `GAMesh` Objektes, die Projektionsebene und die Dimensionsreduktion als Enumeration und wahlweise ein Verschiebungsparameter. Je nach verwendeten Inputparametern führt die Berechnung zu einem der folgenden Ergebnisse.

2D-Triangulation zu 2D-Rechteck Bei Übergabe eines zweidimensionalen Oberflächennetzes in Form eines `GAMesh2D` Objektes wird dessen achsenorientiertes Hüllrechteck berechnet. Der Algorithmus setzt bei voraus, dass die zweidimensionale Triangulation in einer der drei Raumebenen liegt. Im ersten Schritt findet eine Iteration über alle Eckpunkte des Netzes statt, wobei die globalen minimalen und maximalen Koordinaten zwischengespeichert werden. Aus diesen lassen sich anschließend die Eckpunkte des Rechtecks berechnen. Aufgrund der grundsätzlichen Konvexität von Rechteckgeometrien ist es ausreichend nur die Eckpunkte des Netzes zu berücksichtigen, die Kanten können dabei vernachlässigt werden. Dies gilt auch für die nachfolgenden Varianten *4.2.2 3D-Triangulation zu 2D-Projektions-Rechteck* und *4.2.2 3D-Triangulation zu 2D-Querschnitts-Rechteck*. Die Ebene des erzeugten Rechteckes ist die gleiche wie die durch das Netz implizit vorgegebene.

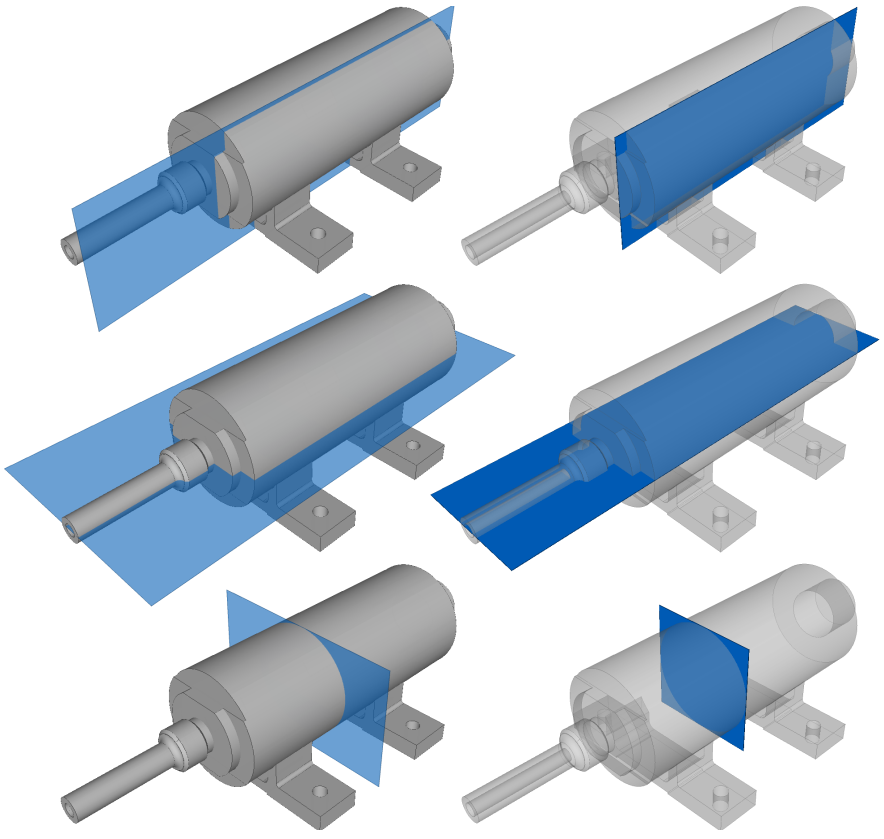


Abbildung 4.9: Achsenorientiertes Hüllrechteck für einen Druckwandler für Raumfahrtantriebe. Links: DimensionReduction. SLICED. Rechts: DimensionReduction. PROJECTED. V.o.n.u.: Ebenen ProjectionPlane.YZ, ProjectionPlane.XY, ProjectionPlane.XZ.

3D-Triangulation zu 2D-Querschnitts-Rechteck Bei Übergabe eines dreidimensionalen Oberflächennetzes in Form eines `GAMesh` Objektes und des Parameters DimensionReduction.SLICED wird das achsenorientierte Hüllrechteck näherungsweise um den Querschnitt in dieser verschobenen Ebene erstellt. Der entwickelte Algorithmus berechnet alle Schnittpunkte der Triangulation in der gegebenen Ebene und speichert diese in einer Punktmenge ab. Das zweidimensionale Hüllrechteck wird in der gegebenen Ebene um die Punktmenge herum erstellt. Auf der rechten Seite von Abb. 4.9 sind Ergebnisse für die Anwendung des 2D-Querschnitts-Rechtecks auf einen Druckwandler für Raumfahrtantriebe für alle drei Projektionsebenen mit einer zur Darstellung passenden Schnittebene dargestellt.

3D-Triangulation zu 2D-Projektions-Rechteck Bei Übergabe eines dreidimensionalen Oberflächennetzes in Form eines `GAMesh` Objektes und des DimensionReduction.PROJECTED Parameters wird unter Berücksichtigung der gewünschten Projektionsebene das achsenorientierte Hüllrechteck näherungsweise um die Projektion des Netzes in dieser Ebene gebildet. Der entwickelte Algorithmus projiziert alle Eckpunkte der Triangulation auf die vorgegebene Ebene und bildet ein zweidimensionales Hüllrechteck um die sich ergebende Punktmenge. Auf der linken Seite von Abb. 4.9 sind Ergebnisse für die Anwendung des 2D-Projektions-Rechtecks auf einen Druckwandler für Raumfahrtantriebe für alle drei Projektionsebenen mit einer zur Darstellung passenden (und der rechten Seite entsprechenden) Verschiebung dargestellt.

Achsenorientierte Pixelisierung

PixelSet
pixelSize : doule[ ] pixelsPerAxis : int[ ] pixelStates : boolean[ ][ ] pixelCenters : double[ ][ ][ ]
PixelSet(input : GAPlanarMesh, disc : int[ ]) PixelSet(input : GAMesh, disc : int[ ], plane : ProjectionPlane, dim : DimensionReduction, type : FillingType, shift : double)

Die Klasse `PixelSet` dient zur Erstellung einer Pixelisierung in zwei Dimensionen. Übergabeparameter sind ein zweidimensionales oder dreidimensionales trianguliertes Oberflächennetz, die Projektionsebene als Enumeration, die Art der Dimensionsreduktion als Enumeration, die Diskretisierung in die beiden die Projektionsebene aufspannenden Raumrichtungen und wahlweise ein Verschiebungsparameter. Wie auch bei der Voxelisierung kann über die Enumerationswerte FillingType.CENTER, FillingType.CORNERS, FillingType.INSIDERECTANGULAR, FillingType. SQUAREDDISTANCE oder FillingType.BYCOLLISION die Berechnungsart des Pixelstatus vorgegeben werden. Je nach gewünschtem Ziel der Geometrievereinfachung führt die Berechnung zu einem der folgenden Ergebnisse.

2D-Triangulation zu Pixelisierung Bei Übergabe eines zweidimensionalen Oberflächennetzes wird dessen Pixelisierung in der durch das Netz implizit vorgegebenen Ebene berechnet. Der Algorithmus setzt bei voraus, dass die zweidimensionale Triangulation in einer der drei Raumebenen liegt. In einem ersten Schritt wird ein zweidimensionales Raster gemäß der vorgegebenen Diskretisierung erzeugt, welches die Koordinaten für die Mittelpunkte der einzelnen Pixel vorgibt. In einer anschließenden geschachtelten Iteration über die Dreiecke der Triangulation und über das Raster wird berechnet, ob die Pixel gefüllt oder leer sind, was abschließend in der Pixelisierung abgespeichert wird.

3D-Triangulation zu 2D-Projektions-Pixelisierung Bei Übergabe eines dreidimensionalen Oberflächennetzes in Form eines `GAMesh` Objektes und des DimensionReduction.PROJECTED Parameters wird unter Berücksichtigung der gewünschten Projektionsebene die Pixelisierung um die Projektion des Netzes gebildet. Der entwickelte Algorithmus ist vergleichbar mit der Pixelisierung bei Übergabe eines zweidimensionalen Netzes, zuvor wird allerdings das dreidimensionale Netz in ein zweidimensionales überführt, indem die implizit über die Enumeration ProjectionPlane ausgewählte Koordinate weggelassen wird. Ist ein Verschiebungsparameter gegeben, wirkt er sich nicht auf die Pixelstatus aus, sondern verschiebt die Pixelisierung lediglich um das gewünschte Maß entlang der Normalen der Pixelisierungsebene. Auf der linken Seite von Abb. 4.10 sind Ergebnisse für die Anwendung der 2D-Projektions-Pixelisierung für alle Projektionsebenen dargestellt.

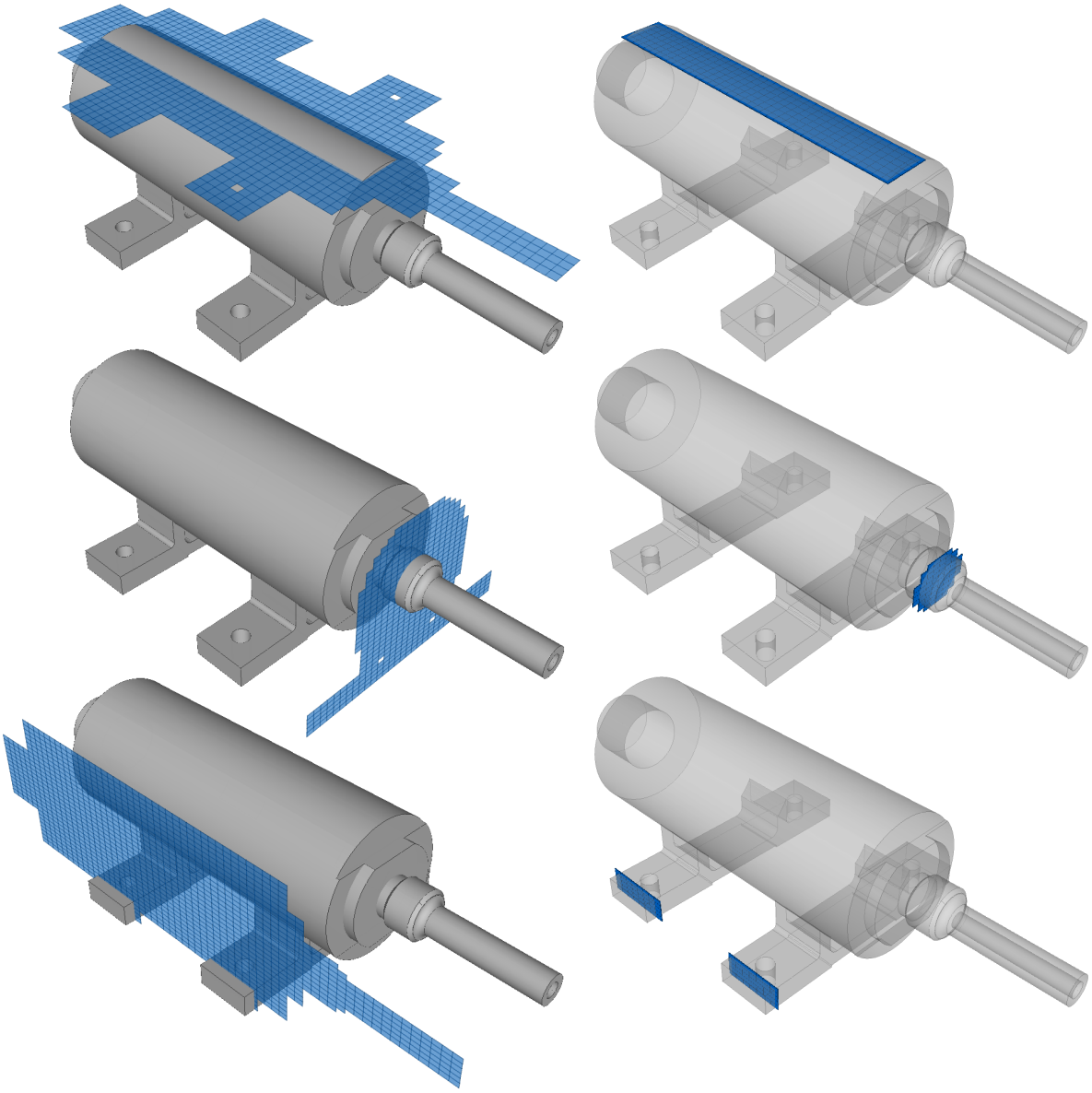


Abbildung 4.10: Varianten der 2D-Querschnitts-Pixelisierung am Beispiel eines Druckwandlers für Raumfahrtantriebe. Links: DimensionReduction.PROJECTED. Rechts: DimensionReduction.SLICED. V.o.n.u.: Ebenen ProjectionPlane.XY, ProjectionPlane.XZ, ProjectionPlane.YZ.

3D-Triangulation zu 2D-Querschnitts-Pixelisierung Bei Übergabe eines dreidimensionalen Oberflächennetzes in Form eines `GAMesh` Objektes, des Parameters DimensionReduction.SLICED, einer gewünschten Ebene und eines Verschiebungsparameters wird die achsenorientierte Pixelisierung aus dem Querschnitt der Triangulation in dieser verschobenen Ebene erstellt. Der entwickelte Algorithmus ist vergleichbar mit der Pixelisierung bei Übergabe eines zweidimensionalen Netzes, das zweidimensionale Raster wird allerdings in der verschobenen Ebene mit dem Verschiebungsparameter als dritte Koordinate erstellt und die Überprüfung der Pixelfüllung findet in drei Dimensionen statt. Auf der rechten Seite von Abb. 4.10 sind Ergebnisse für die Anwendung der 2D-Querschnitts-Pixelisierung auf einen Druckwandler für Raumfahrtantriebe für alle drei Projektionsebenen mit einer zur Darstellung passenden (und der linken Seite entsprechenden) Verschiebung dargestellt.

Achsenorientierter Hüllrechteckverbund

BoundingRectangleSet
rectangles : AxisAlignedBoundingRectangle[ ]
BoundingRectangleSet(input : PixelSet, order : int[ ], plane : ProjectionPlane, dim : DimensionReduction, shift : double) BoundingRectangleSet(input : PixelSet, number : int, optiType : OptimizationType, plane : ProjectionPlane, dim : DimensionReduction, shift : double)

Die Klasse `BoundingRectangleSet` dient zur Erstellung eines sog. achsenorientierten Hüllrechteckverbundes in zwei Dimensionen. Übergabeparameter sind die Pixelisierung eines zweidimensionalen oder dreidimensionalen triangulierten Oberflächennetzes und je nach ausgewähltem Verfahren eine Vereinigungsreihenfolge oder eine Rechteckanzahl sowie wahlweise ein Optimierungsverfahren. Die Pixelisierung (siehe *4.2.2 Achsenorientierte Pixelisierung*) bildet die Basis für die Erzeugung des Hüllrechteckverbundes, wobei die Qualität des Verbundes in erster Linie von deren Qualität abhängt. Die entwickelte Entwurfssprache bietet zwei mögliche Verfahren zur Erzeugung von Hüllrechteckverbunden, den Verbund durch Pixel-Zusammenfassen und den optimierten Verbund mit einer vorgegebenen Anzahl von Rechtecken. Beide Algorithmen ergeben sich aus den jeweiligen Algorithmen in drei Dimensionen (*4.2.2 Optimierter Quaderverbund aus Paketierung* und *4.2.2 Optimierter Quaderverbund aus Voxelisierung*) durch Weglassen der nicht benötigten Koordinate.

Planare konvexe Hülle

PlanarConvexHull
hull : GAPlanarMesh
PlanarConvexHull(input : PlanarGAMesh) PlanarConvexHull(input : GAMesh, plane : ProjectionPlane, dim : DimensionReduction, shift : double)

Die Klasse `PlanarConvexHull` dient zur Erstellung einer zweidimensionalen planaren konvexen Hülle in Form eines Polygons. Übergabeparameter sind ein zweidimensionales oder dreidimensionales trianguliertes Oberflächennetz, die Projektionsebene als Enumeration, die Art der Dimensionsreduktion als Enumeration und wahlweise ein Verschiebungsparameter.

2D-Netz zu planarer Hülle Bei Übergabe eines zweidimensionalen Oberflächennetzes wird dessen planare konvexe Hülle in der durch das Netz implizit vorgegebenen Ebene mithilfe der

`vtkDelaunay2D` Klasse aus der *VTK* Bibliothek [Schroeder et al., 2006] berechnet. Der Algorithmus setzt bei voraus, dass die zweidimensionale Triangulation in einer der drei Raumebenen liegt.

3D-Netz zu 2D-Projektions-Hülle Bei Übergabe eines dreidimensionalen Oberflächennetzes in Form eines `GAMesh` Objektes und des Parameters DimensionReduction.PROJECTED wird unter Berücksichtigung der gewünschten Projektionsebene eine planare Konvexe Hülle um die Projektion des Netzes gebildet. Der entwickelte Algorithmus projiziert zunächst alle Eckpunkte der Triangulation auf die gewünschte Ebene, welche anschließend auf eine x-y-Ebene transformiert wird, in welcher die Berechnung der planaren Hülle mithilfe der `vtkDelaunay2D` Klasse aus der *VTK* Bibliothek [Schroeder et al., 2006] stattfindet. Die erzeugte Hülle wird im Anschluss wieder in die Ausgangsebene rücktransformiert.

3D-Netz zu 2D-Querschnitts-Hülle Bei Übergabe eines dreidimensionalen Oberflächennetzes in Form eines `GAMesh` Objektes, des Parameters DimensionReduction.SLICED, einer gewünschten Ebene und eines Verschiebungsparameters wird die planare konvexe Hülle aus dem Querschnitt der Triangulation in dieser verschobenen Ebene erstellt. Der entwickelte Algorithmus schneidet hierfür die Triangulation und extrahiert alle Schnittpunkte. Diese werden in eine Ebene transformiert, in der die konvexe Hülle mithilfe der `vtkDelaunay2D` Klasse aus der *VTK* Bibliothek [Schroeder et al., 2006] berechnet wird. Es folgt eine Rücktransformation in die ursprüngliche Ebene.

Hüllkreis

BoundingCircle
center : double[ ] radius : double
BoundingCircle(input : GAPlanarMesh) BoundingCircle(input : GAMesh plane : ProjectionPlane, dim : DimensionReduction, shift : double)

Die Klasse `BoundingCircle` dient zur Erstellung eines Hüllkreises in zwei Dimensionen. Übergabeparamter sind ein zweidimensionales oder dreidimensionales trianguliertes Oberflächennetz, die Projektionsebene als Enumeration und wahlweise ein Verschiebungsparameter sowie der Dimensionsreduktionstyp als Enumeration. Je nach verwendeten Inputparametern führt die Berechnung zu einem der folgenden Ergebnisse.

2D-Netz zu Hüllkreis Bei Übergabe eines zweidimensionalen Oberflächennetzes wird dessen zweidimensionaler Hüllkreis berechnet. Dabei gibt es keine Einschränkungen bezüglich der Ausrichtung der durch das Oberflächennetz implizit vorgegebenen Ebene im Raum. Der Algorithmus greift auf die `BoundingSphere` Klasse zu und übergibt alle im Oberflächennetz enthaltenen Eckpunkte zur Berechnung einer Hüllkugel. Aus dieser können der Mittelpunkt und der Radius für den Hüllkreis übernommen werden. Die Ebene des erzeugten Hüllkreises ist die gleiche wie die durch das Netz implizit vorgegebene.

3D-Netz zu 2D-Projektions-Hüllkreis Bei Übergabe eines dreidimensionalen Oberflächennetzes und des Parameters DimensionReduction.PROJECTED wird unter Berücksichtigung der gewünschten Projektionsebene der Hüllkreis näherungsweise um die Projektion des Netzes in dieser Ebene gebildet. Der entwickelte Algorithmus projiziert alle Eckpunkte der Triangulation

auf die vorgegebene Ebene und bildet einen zweidimensionalen Hüllkreis um die sich ergebende Punktmenge wie in *4.2.2 2D-Netz zu Hüllkreis* beschrieben.

3D-Netz zu 2D-Querschnitts-Hüllkreis Bei Übergabe eines dreidimensionalen Oberflächennetzes und des Parameters DimensionReduction.SLICED wird der Hüllkreis näherungsweise um den Querschnitt in dieser verschobenen Ebene erstellt. Der entwickelte Algorithmus berechnet alle Schnittpunkte der Triangulation in der gegebenen Ebene und speichert diese in einer Punktmenge ab. Der zweidimensionale Hüllkreis wird in der gegebenen Ebene um die Punktmenge herum erstellt, wie in *4.2.2 2D-Netz zu Hüllkreis* beschrieben.

Eigenschaften Achsenorientierter Geometrieannäherungen

Die Güte aller achsenorientierten Geometriedarstellungen ist vom eingehenden Oberflächennetz und dessen Orientierung im Raum abhängig. Bei langen geraden Kanten oder Flächen wird empfohlen, das Netz zugunsten einer höheren Qualität vorher zu rotieren, sodass die längsten Kanten bzw. Flächen parallel bzw. koplanar zu den Achsen bzw. Ebenen liegen. Dadurch lassen sich bessere Ergebnisse bezüglich Genauigkeit und Anzahl der benötigten vereinfachenden Entitäten (wie Rechtecke bzw. Pixel oder Boxen bzw. Voxel) erzielen.

In der Packing-Optimierung bietet die achsenorientierte Geometrie-Approximation sowohl in 2D als auch in 3D den Vorteil, dass die Anwendung von einfachen linearen Formulierungen für die Kollisionserkennung und -vermeidung sowie für das geometrische Enthaltensein ermöglicht wird. Dies gilt sowohl für das achsenorientierte Hüllrechteck bzw. den Hüllquader als auch für die achsenorientierte Pixelisierung bzw. Voxelisierung, wie auch für die darauf aufbauenden Vereinfachungen wie der achsenorientierte Hüllrechteckverbund und den achsenorientierten Verbund aus quaderförmigen Hüllkörpern. Eine nähere Beschreibung und Diskussion zu dieser Thematik folgt in *4.4.3 Nebenbedingungen und Zwänge*. In diesem Kontext sei auch darauf hingewiesen, dass bei Verwendung einer achsenorientierten Geometrieapproximation nicht zwingend auf rotatorische Freiheitsgrade verzichtet werden muss. Bei der entwickelten Packing-Optimierung kann ein diskreter rotatorischer Freiheitsgrad eines Objektes durch binäre rotatorische Zustände und zugehörige achsenorientierte Geometrienäherungen repräsentiert werden. Eine Beschreibung dessen folgt in *4.4.1 Repräsentation der Freiheitsgrade über Szenarien*.

4.3 Optimierung im Packing Framework

Aus der Notwendigkeit für automatisierte Packingprozesse entstanden, aber dennoch universell einsetzbar, ist das in der vorliegen Arbeit entwickelte und implementierte Modul zur Optimierung. Im Kontext der vorliegenden Arbeit kommt dieses zwar speziell bei der Lösung von Packingproblemen zum Einsatz, es ist jedoch unabhängig davon gestaltet und kann für jegliche Art von Optimierungsproblem Verwendung finden. Im Folgenden soll dieses flexible Modul vorgestellt werden, welches über eine Ontologie von Interfaces und abstrakten sowie konkreten Klassen verfügt, um die Implementierung von Optimierungsproblemen und deren Lösungsmethoden zu unterstützen. Ausgehend von einer Vorstellung der Metaontologie des Optimierungsframeworks wird in *4.3.1 Optimierungsverfahren als Entwurfssprache* zunächst auf die Möglichkeiten der Umsetzung einer Optimierung als Entwurfssprache eingegangen. Nachfolgend und darauf aufbauend wird in *4.3.2 Optimierungsproblem als Entwurfssprache* die Ontologie für die Modellierung von Optimierungsproblemen präsentiert, gefolgt von einer Diskussion der gegenseitigen Abhängigkeit von Optimierungsproblem und Lösungsverfahren in *4.3.3 Interdependenz der Problemdarstellung und -lösung.* Abschließend werden die entwickelten Verfahren zur Lösung von Optimierungsproblemen vorgestellt, was zum einen die Entwicklung einer Schnittstelle an bestehende Optimierungssoftware in *4.3.4 Mathematische Optimierung als Entwurfssprache* einschließt und zum anderen die Implementierung einer eigenständigen Metaheuristik in *4.3.5 Partikel-Multi-Schwarm Optimierung als Entwurfssprache* umfasst. Ferner wurde das vorgestellte Framework im Sinne eines Funktionsnachweises in mehreren akademischen Anwendungsfällen eingesetzt, welche in Kapitel 5 zu finden sind.

Das entwickelte Optimierungsframework besteht aus zwei Hauptbestandteilen, einer Entwurfssprache für die Problemdefinition und einer Entwurfssprache für die Problemlösung. Der Lösungsprozess findet mithilfe eines Optimierungsalgorithmus oder einer Optimierungssoftware statt und ist durch die zweiteilige Frameworkstruktur von der Problemdefinition entkoppelt. Der Workflow beginnt mit der Modellierung des Optimierungsproblems durch den/die Anwender/-in auf Basis der Entwurfssprache zur Problemdefinition. Zur Laufzeit wird das Optimierungsproblem mithilfe des Optimierungsverfahrens in eine von dem Algorithmus oder der Software lesbare Form transformiert und dem Lösungsprozess zur Verfügung gestellt. Das Optimierungsverfahren dient dabei zugleich als Schnittstelle und Lösungsverfahren, d.h. es liest das Problem ein und gibt nach einer spezifizierten Anzahl an Iterationen oder in einem vorgegebenen Zeitrahmen die (sub-)optimale Lösung des Problems zurück. Auf diese Weise wird eine klare Aufgabenteilung der beiden Frameworkmodule erreicht, welche einen entscheidenden Vorteil mit sich bringt: Verschiedene Optimierungsalgorithmen oder -softwares können leicht an demselben Problem getestet und miteinander verglichen werden, andersherum kann ein bestimmter Optimierungsalgorithmus bzw. eine spezielle Software aber auch für verschiedene Optimierungsprobleme zum Einsatz kommen.

In Abbildung 4.11 ist die Ontologie der Optimierungsentwurfssprache als UML-Klassendiagramm dargestellt. Die Optimierung besteht aus den beiden Hauptkomponenten Optimierungsproblem (`Problem`) und Optimierungsverfahren (`OptimizationProcess`), umgesetzt durch einen speziellen Optimierungsalgorithmus oder die Anbindung zu einer Optimierungssoftware als erbende Klassen. Das berechnete Ergebnis (`Solution`) ist als autonome Klasse spezifiziert. Für die Anwendung der Optimierungs-Entwurfssprache muss zur Problemdefinition das Paket `optimization` in die zu optimierende Entwurfssprache importiert werden. Die Ontologie der Problemdefinition wird in *4.3.2 Optimierungsproblem als Entwurfssprache* näher erläutert. Dort wird auch auf die verschiedenen Problemtypen (`ProblemType`) näher eingegangen.

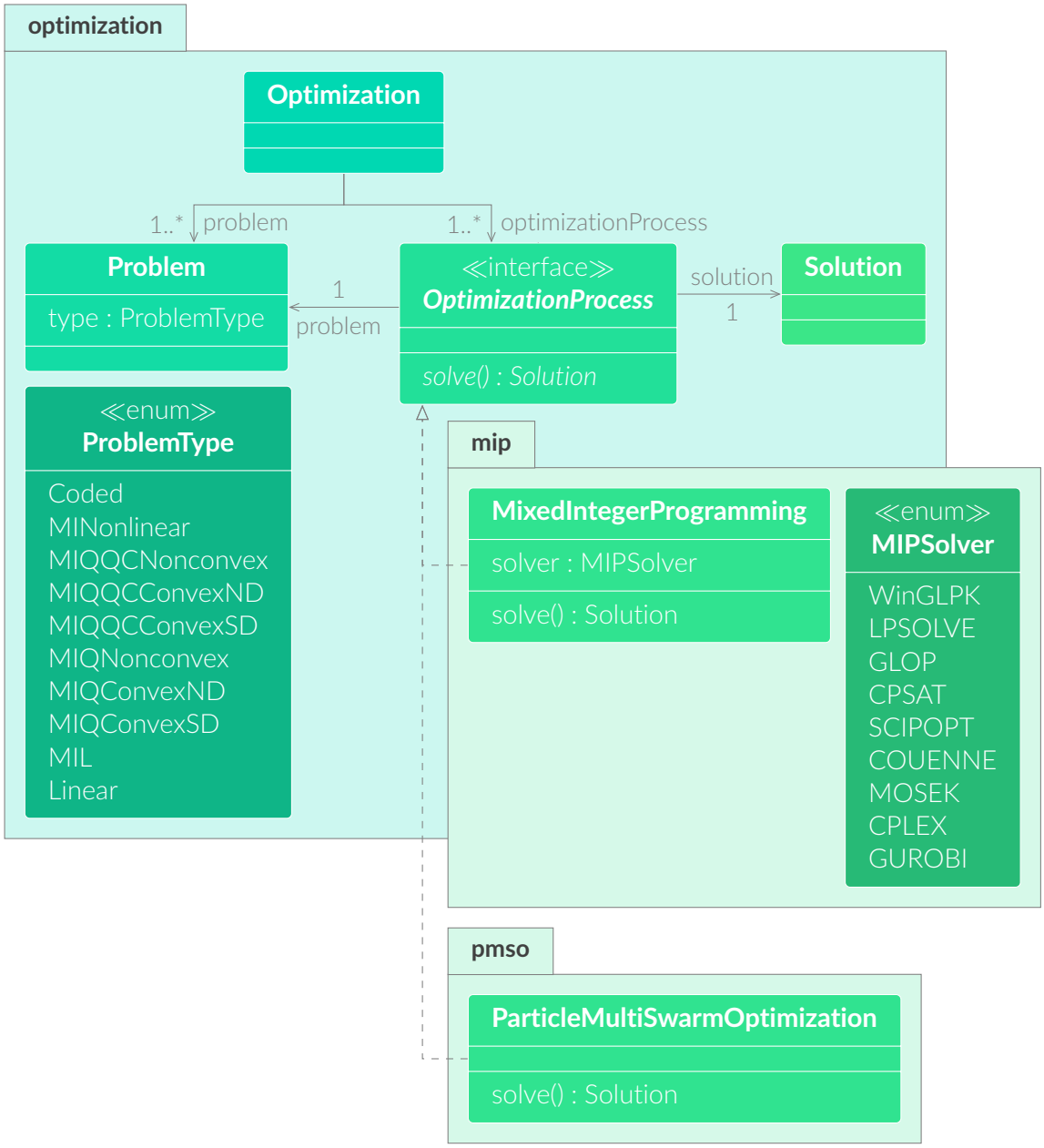


Abbildung 4.11: Ontologie der Optimierungsentwurfssprache als UML-Klassendiagramm.

Zusätzlich muss je nach gewünschtem Optimierungsverfahren das jeweilige zugehörige Paket importiert werden. Ist ein mathematisches Optimierungsverfahren gewünscht, kann die gemischt-ganzzahlige lineare oder nichtlineare Optimierung (`MixedIntegerProgramming`) aus dem Paket `mip` eingesetzt werden. Wird stattdessen eine metaheuristische Optimierung bevorzugt, kann die Partikel-Multi-Schwarm-Optimierung (`ParticleMultiSwarmOptimization`) aus dem Paket `pmso` zum Einsatz kommen. Durch die Aufspaltung in einzelne Pakete können die zugehörigen Optimierungsalgorithmen unabhängig voneinander zur Verfügung gestellt und importiert werden. Diese Aufspaltung dient zum einen der einfachen Erweiterbarkeit, soll zum anderen aber auch vor zu großen Importdateien und damit vor unnötig hohen Datenmengen schützen. Die vorgestellten Optimierungsverfahren werden im Folgenden in *4.3.1 Optimierungsverfahren als Entwurfssprache* genauer diskutiert. Dort wird auch auf die verschiedenen Möglichkeiten ihrer Integration in eine Entwurfssprache eingegangen. Eine Beschreibung der

verschiedenen Optimierungsprogramme (`MIPSolver`) findet sich in *4.3.4 Mathematische Optimierung als Entwurfssprache.*

4.3.1 Optimierungsverfahren als Entwurfssprache

Soll das Stichwort Optimierung in Entwurfssprachen integriert werden, steht am Anfang der Überlegungen die Auswahl einer geeigneten Integrationsstrategie. Im Allgemeinen gibt es verschiedene Vorgehensweisen, um den Optimierungsprozess zu implementieren, in der vorliegenden Arbeit wurden drei unterschiedliche Varianten entwickelt, implementiert und untersucht. Die Untersuchung der drei grundlegend verschiedenen Varianten ist zunächst von dem Bestreben geleitet, einen umfassenden Überblick zu gewinnen, um die Eignung für ein universelles Optimierungsframework zu prüfen. Ein weiteres Argument für die Integration mehrerer Methoden ist vor dem Hintergrund eines ganzheitlichen Ansatzes zu nennen, für den ein Framework geschaffen werden soll, welches flexibel genug ist, verschiedene Vorgehensweisen zu unterstützen. Dadurch soll die Möglichkeit geschaffen werden, unterschiedliche Methoden zu kombinieren und zu vereinen, um noch bessere Gesamtansätze zu finden. Die in der vorliegenden Arbeit integrierten Ansätze sollen im Folgenden vorgestellt werden.

Optimierungsalgorithmus als Entwurfssprache Die erste Strategie für die Integration von Optimierungsverfahren in graphenbasierte Entwurfssprachen ist die Implementierung eines vollständigen Optimierungsalgorithmus als eigenständige Entwurfssprache. Dafür muss die gesamte Ontologie des Algorithmus aufgebaut und in Form eines UML-Klassendiagramms dem Benutzer zur Verfügung gestellt werden. Wird ein autonomer Optimierungsalgorithmus auf diese Weise implementiert, kann dieser in andere Entwurfssprachen importiert und so zur Verwendung bereitgestellt werden. Die Iteration während des Optimierungsprozesses findet dabei innerhalb des Produktionssystems der Optimierungsentwurfssprache statt. In Abb. 4.12 ist der Optimierungsprozess schematisch dargestellt. Die benötigte Ontologie ist in hellgrün hervorgehoben. Der Hauptvorteil dieser Vorgehensweise ist die Unabhängigkeit von anderer Software. Wird zusätzlich ein Algorithmus gewählt, welcher in der Lage ist, neben mathematischen Funktionen auch Methoden in Form von Programmcode oder sogar ganze Simulationsprogramme auszuwerten, kann jede erdenkliche (auch bereits implementierte) Funktion als Rand- oder Nebenbedingung in das Optimierungsproblem mit eingehen. Solche Algorithmen sind beispielsweise durch heuristische und metaheuristische Optimierungsverfahren gegeben. Für die Schaffung eines Frameworks ist das metaheuristische Verfahren allerdings dem heuristischen vorzuziehen, da es theoretisch auf beliebige Problemstellungen angewandt werden kann und erst bei konkreter Anwendung durch den/die Nutzer/-in problemspezifisch implementiert werden muss. Das macht metaheuristische Verfahren zur idealen Basis für die Bereitstellung von unterstützenden Bibliotheken. Es ist hierbei zu erwähnen, dass das Auffinden einer globalen optimalen Lösung bei der Verwendung von Metaheuristiken im Regelfall nicht garantiert werden kann. Zwar haben solche Verfahren in der Literatur bereits bewiesen, dass sie gute Lösungen liefern, diese können aber theoretisch auch beliebig schlecht im Vergleich zum globalen Optimum sein. Der Erfolg und die Laufzeit hängen wesentlich von der konkreten Implementierung ab. Die direkte Umsetzung der Optimierung als metaheuristischer Algorithmus ermöglicht aber eine einfache Verwendung von bereits vorhandenen Prozeduren und Methoden, selbst wenn sie lediglich als Blackbox Funktionen vorliegen. Ein weiterer Vorteil ist, dass die bestehenden Modelle wenig angepasst werden müssen, da lediglich die Optimierungsvariablen als solche gekennzeichnet werden müssen und es keine spezielle Vorgabe für die Problemrepräsentation gibt. Dadurch können ganze Teilprogramme in Entwurfssprachen optimiert werden, ohne dass

Ersatzprobleme modelliert werden müssen. Ein Nachteil dieser Methode liegt im potenziellen Implementierungsaufwand, da die Optimierungsalgorithmen nachprogrammiert werden müssen, wenn sie nicht bereits in der passenden Programmiersprache vorliegen. Sollen verschiedene Verfahren getestet und gegeneinander abgewogen werden, kann der Umfang des umzusetzenden Codes mitunter sehr groß werden. Bei einer ausreichend umfassenden Verfügbarkeit von Optimierungsbibliotheken, kann dieser Punkt jedoch vernachlässigt werden. Insgesamt liegt der Fokus bei dieser Vorgehensweise weniger auf Seiten der Problemmodellierung und viel mehr auf Seiten der Entwicklung und Implementierung von Optimierungsverfahren oder -bibliotheken. Ist ein oder sind mehrere geeignete Verfahren erst mal implementiert, verschiebt sich der Fokus zur Effizienzsteigerung der Optimierungsalgorithmen. Dabei muss berücksichtigt werden, dass hierfür ein umfangreiches Expertenwissen im Bereich der Optimierungsalgorithmik notwendig ist. In der vorliegenden Arbeit wird zur Repräsentation dieser Variante eine Partikel-Multi-Schwarm-Optimierung ausgewählt. Die zugehörige Entwurfssprache wird in *4.3.5 Partikel-Multi-Schwarm Optimierung als Entwurfssprache* detailliert vorgestellt.

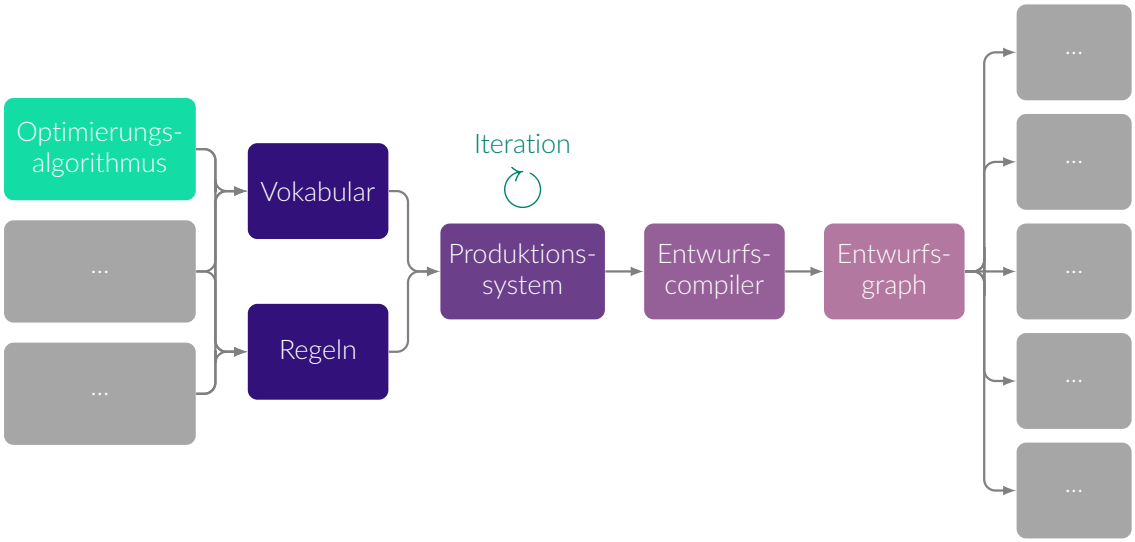


Abbildung 4.12: Eigenständiger Optimierungsalgorithmus als Entwurfssprache

Entwurfssprache als Schnittstelle zur Optimierungssoftware Eine zweite Strategie für die Integration von Optimierungsverfahren in die Entwurfssprachen ist die Anbindung von bereits bestehenden Optimierungstools. Diese Art der Umsetzung bietet sich vor allem beim Einsatz von globalen mathematischen Optimierungsverfahren an, da hierfür eine Reihe geeigneter und direkt einsetzbarer Lösungen in Form von freien oder kommerziellen Programmen zur Verfügung steht. Globale mathematische Verfahren haben einen großen Vorteil gegenüber heuristischen oder metaheuristischen Verfahren. Sie sind dafür ausgelegt bei ausreichender Rechenzeit die tatsächliche globale optimale Lösung zu finden. Soll nun externe Software verwendet werden, muss eine Ontologie zum Ansprechen der Software über eine Programmierschnittstelle geschaffen und dem Nutzer über ein UML-Klassendiagramm bereitgestellt werden. Für den Optimierungsprozess muss außerdem die jeweilige Optimierungssoftware zur Verfügung stehen. Wird die Ontologie für Optimierungssolver in eine andere Entwurfssprache importiert, können die Optimierungsmodelle auf dieser Basis vom Nutzer aufgebaut und an den jeweiligen Solver zur Lösung übergeben werden. Die Iteration während des Lösungsprozesses findet in den angebundenen Optimierern statt. In Abbildung 4.13 ist dieser Optimierungsprozess schematisch dargestellt. Die benötigten Anteile wie die Ontologie und die externen Optimierer

sind in hellgrün hervorgehoben. Der Vorteil dieser Methode liegt darin, dass kein explizites Expertenwissen über effiziente Optimierung und deren Lösungsverfahren vorliegen muss, da der Lösungsvorgang entkoppelt innerhalb der Software stattfindet. Die Expertise in und der Implementierungsaufwand von Optimierungsverfahren kann so an die Softwarehersteller ausgelagert werden. Da auf den Lösungsprozess nur wenig Einfluss genommen werden kann, liegt der Fokus bei dieser Vorgehensweise auf der Modellierung von effizienten Ersatzproblemen. Die Verwendung von Optimierungssoftware setzt voraus, dass das Optimierungsproblem in einer geeigneten Repräsentationsform definiert und zur Laufzeit an die Optimierungssoftware übergeben wird. Bei klassischen Solvern für globale mathematische Optimierung z.B. ist diese spezielle Problemdarstellung ein mathematisches System von Gleichungen und Ungleichungen. Dies erfordert zum einen Erfahrung, da die geeignete Problembeschreibung von dem gewünschten Optimierer abhängt, zum anderen muss genug Wissen über die Komplexität der zu verwendenden mathematischen Funktionen vorhanden sein. Diese Herausforderung stellt auch den Nachteil bei der vorgestellten Strategie dar, da die Gestaltung effizienter Modelle je nach geforderten Rand- und Nebenbedingungen mitunter hochkomplex sein kann. Werden mathematische Optimierer aufgerufen, ergibt sich außerdem ein weiterer Nachteil. Die meisten Programme bringen die Voraussetzung mit, dass nur mathematische Repräsentationen verwendet werden dürfen, was es schwierig macht, bereits vorhandene Methoden oder Blackbox-Funktionen als Nebenbedingungen innerhalb der Optimierung zu verwenden. Ein typisches Beispiel für solche Blackbox Funktionen sind Simulationen, welche in der Optimierungsschleife eingehen sollen. Zuletzt ist die potenziell notwendige Lizenznahme bei proprietärer Software als weiteres Manko zu nennen, welches entfällt, wenn ein Programm sowieso bereits lizenziert wird oder eine Open-Source-Software bzw. eine Freeware Alternative angebunden wird. In der vorliegenden Arbeit wurde eine Ontologie für die Anbindung von einer Reihe von sowohl proprietären als auch freien globalen mathematischen Optimierungsprogrammen entwickelt und als Entwurfssprache umgesetzt. Diese wird in *4.3.4 Mathematische Optimierung als Entwurfssprache* detailliert besprochen.

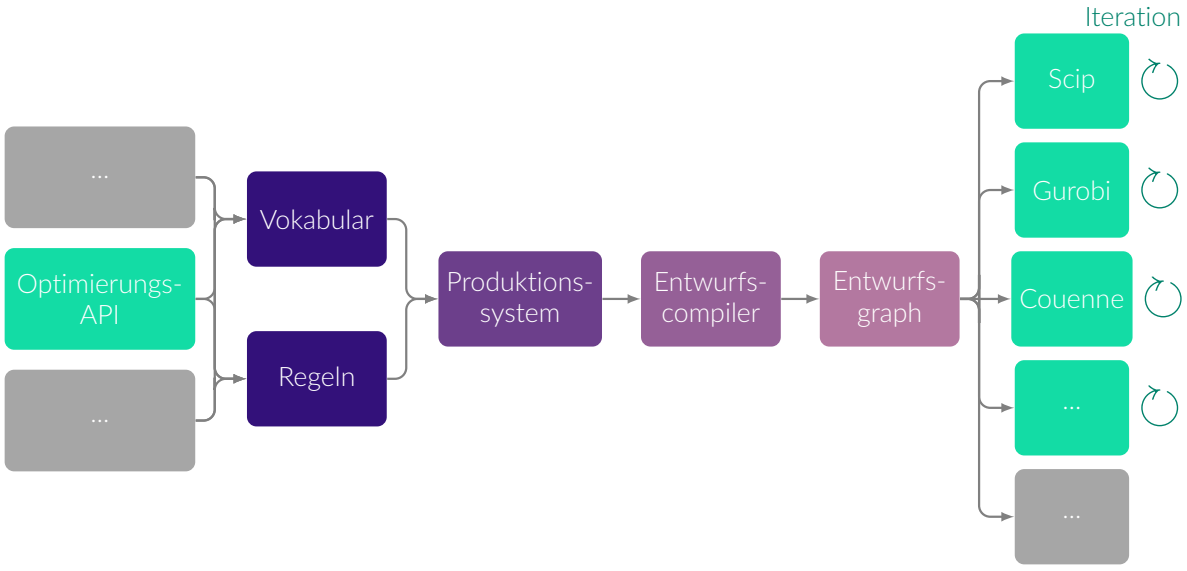


Abbildung 4.13: Entwurfssprache als Schnittstelle zur Optimierungssoftware

Aufruf der Entwurfssprache aus einer Optimierungssoftware Neben den beiden bereits genannten Möglichkeiten existiert eine dritte Variante zur Integration der Optimierung in Entwurfssprachen. In dieser wird eine Optimierungsplattform verwendet, welche in jedem Itera-

tionsschritt der Optimierung die zu optimierenden Entwurfssprachen selbst aufruft. Abbildung 4.14 stellt den Workflow bei dieser Vorgehensweise dar. Zur Definition der Variablen kann eine textbasierte Austauschdatei angelegt werden, welche innerhalb der Entwurfssprache mit den berechneten Werten für die jeweiligen Variablen befüllt wird. Die Rand- und Nebenbedingungen können dabei implizit in der Entwurfssprache enthalten sein und müssen in Form einer Ersatzvariable ebenfalls in die Datei geschrieben werden. Die Optimierung selbst findet unabhängig von den Entwurfssprachen in der Optimierungssoftware statt. Der Vorteil dieser Methode ist der geringe Implementierungsaufwand, da lediglich die Austauschdatei spezifiziert werden muss. Der Hauptnachteil liegt in der Anwendung der Software selbst, da effiziente Plattformen häufig einer potenziell kostenintensiven Lizenz bedürfen. In der vorliegenden Arbeit wird diese Variante lediglich der Vollständigkeit halber im Anwendungsbeispiel in *5.2 Packing in einem Flugzeugtragflügel* eingesetzt, sie wird jedoch insgesamt als ungeeignet für ein Optimierungsframework eingestuft und soll im Folgenden nur am Rande betrachtet werden.

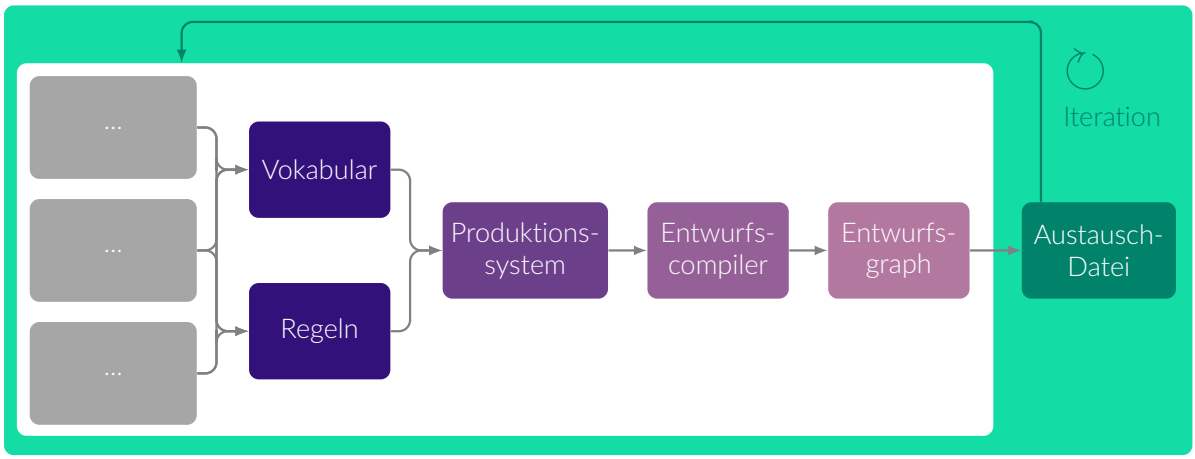


Abbildung 4.14: Aufruf der Entwurfssprache aus einer Optimierungssoftware

4.3.2 Optimierungsproblem als Entwurfssprache

Wie zu Anfang des Kapitels Abschnitt 4.3 bereits erwähnt, ist eines der beiden vorgestellten Frameworkmodule die Entwurfssprache zur Problemdefinition. Im Folgenden soll diese detailliert vorgestellt werden, wobei die Ontologie der Entwurfssprache deren Kern bildet. Diese Ontologie beinhaltet alle Komponenten, welche zur Problemdefinition durch den/die Anwender/-in benötigt werden. Ein Problem (siehe Klasse `Problem` in Abb. 4.15) benötigt in erster Linie mindestens eine Variable (siehe Klasse `Variable` in Abb. 4.15) und mindestens eine Restriktion (siehe Klasse `Constraint` in Abb. 4.15) oder eine Zielfunktion (siehe Klasse `Fitness` in Abb. 4.15), welche die Variablenwerte verarbeiten. Im Allgemeinen kann ein Problem aus einer nicht begrenzten Anzahl an Restriktionen mit oder ohne Zielfunktion bestehen. Ist keine Zielfunktion definiert, handelt es sich um kein klassisches Optimierungsproblem und es wird statt einer Optimierung lediglich eine Lösung des Gesamtproblems durchgeführt, d.h. die erste valide Lösung, welche gefunden wird, wird am Ende ausgegeben.

Soll ein Optimierungsproblem modelliert werden, müssen zunächst die eingehenden Variablen spezifiziert werden. Dafür muss ein Name, eine untere Schranke, eine obere Schranke und der Variablentyp übergeben werden. Der Variablentyp kann über die Enumeration `VariableType` spezifiziert werden, indem einer der Enumerationswerte an den Konstruktor der `Variable`

Klasse übergeben wird. Der Leser sei an dieser Stelle darauf hingewiesen, dass alle Konstruktoren zur besseren Übersichtlichkeit in Abbildung 4.15 vernachlässigt werden. Der Enumerationswert `BOOL` steht als Abkürzung für *boolean* für eine boolesche Variable, `INT` als Abkürzung für den Datentyp *integer* steht für eine Variable, welche ausschließlich ganze Zahlen annehmen kann und der Enumerationswert `REAL` steht für eine kontinuierliche Variable deren Werte vom Datentyp *double* sind. Weiterhin kann bei der Erstellung einer Variable optional ein Startwert übergeben werden. Dieser gilt dann als Ausgangspunkt für die Optimierung der Variable.

Wird eine Zielfunktion definiert, muss von dem/der Anwender/-in spezifiziert werden, ob es sich um ein Minimierungs- oder Maximierungsproblem handelt. Zur Spezifikation kann die Enumeration `ObjectiveType` verwendet werden und es muss einer der beiden Enumerationswerte an den Konstruktor der `Fitness` Klasse übergeben werden. Es gibt mehrere Zielfunktionsklassen, welche von der abstrakten `Fitness` Klasse erben und deren `CalcFitness(map:Map<Variable, double>)` Methode implementieren. In dieser wird die Zielfunktion zur Laufzeit mit den jeweils aktuell geltenden Variablenwerten ausgewertet. Die Variablen und deren zugehörigen aktuellen Werte werden dabei in Form einer Map an diese Auswertungsmethode übergeben. Es gibt einen grundlegenden Unterschied zwischen der `CodeFitness` Klasse und den übrigen Zielfunktionsklassen (`NonLFitness`, `QuadFitness` und `LinFitness`), welche das `Equational` Interface implementieren. `Equational` steht hierbei dafür, dass es sich bei der Zielfunktion um eine mathematische Funktion handelt. Aus den letzteren Klassen können Objekte direkt instanziiert werden, wobei alle Terme zur Definition der Zielfunktion an den Konstruktor übergeben werden müssen. Soll ein `CodeFitness` Objekt erzeugt werden, muss von dem/der Anwender/-in zunächst eine Unterklasse spezifiziert werden, in der die Auswertungsmethode überschrieben wird. Um dieses Vorgehen sicherzustellen, ist die `CodeConstraint` Klasse in Form einer abstrakten Klasse in der Ontologie angelegt. Diese grundlegende Unterscheidung lässt sich auch auf die Charakteristik der Auswertungsmethode übertragen. Während die Auswertungsmethode eines `CodeFitness` Objektes Programmcode beinhalten und aufrufen kann, sind die Methoden der übrigen Klassen vordefiniert und werten die Zielfunktion durch Einsetzen der aktuellen Variablenwerte in die zugehörigen Terme aus.

Eine Restriktion besteht immer aus drei Komponenten, einer Linkshandseite, in der Ontologie bezeichnet als Bedingung bzw. *Condition*, einem Operator und einer Rechtshandseite. Die Rechtshandseite darf lediglich aus einer Konstanten bestehen, d.h. die Bedingung muss alle an der Restriktion beteiligten Variablen enthalten bzw. verarbeiten. In Konsequenz muss der/die Anwender/-in Restriktionen bei Bedarf vorab umstellen, sodass diese geforderte Form gegeben ist. Alle drei Bestandteile müssen bei der Erstellung eines Restriktionsobjektes an den Konstruktor der jeweiligen Klasse übergeben werden. Der Operator wird dabei über die Enumeration `Operator` spezifiziert, wobei der Enumerationswert `GE` als Abkürzung für *greater equal* für größer gleich steht, der Wert `LE` als Abkürzung für *lower equal* für kleiner gleich steht und `EQ` als Abkürzung für *equal* Gleichheit symbolisiert. Hinsichtlich der Instanziierung gelten bei der Modellierung von Restriktionen dieselben Bedingungen wie für die Modellierung von Zielfunktionen. Auch hier gibt es die Einteilung in die Subklassen `CodeConstraint` und die das `Equational` Interface implementierenden Klassen `NonLConstraint`, `QuadConstraint` und `LinConstraint`, welche die Auswertungsmethode implementieren. `Equational` steht hierbei dafür, dass es sich bei der Restriktion um eine mathematische Gleichung oder Ungleichung handelt. Die Auswertungsmethode `calcCondition(map:Map<Variable, double>)` wertet diesmal die Linkshandseite, also die Bedingung der Restriktion aus. Sie muss bei Instanziierung eines von der `CodeConstraint` Klasse erbenden Objektes von dem/der Anwender/-in vorab in einer Subklasse spezifiziert werden. Bei Instanziierung der übrigen Restriktionen müssen diese

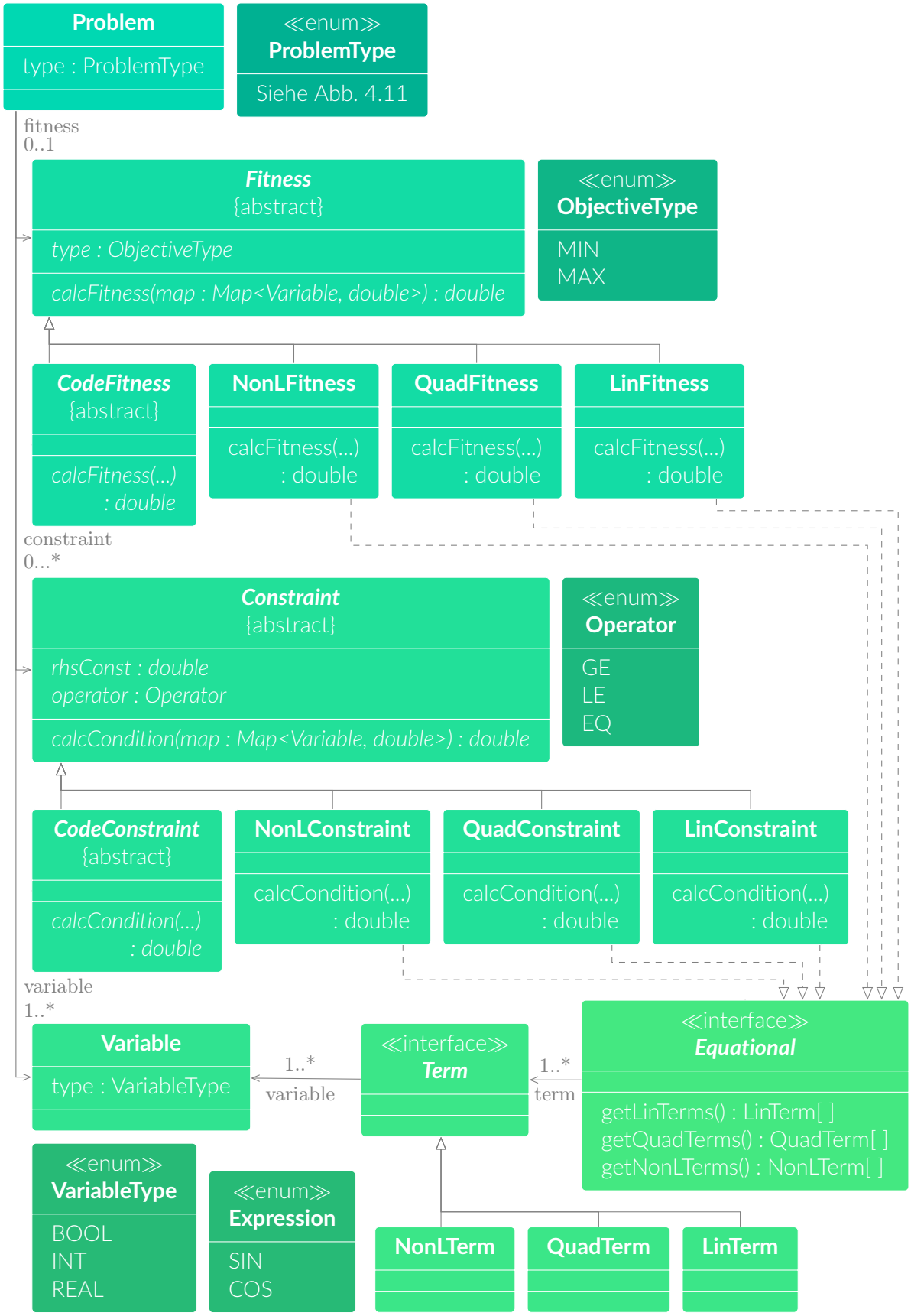


Abbildung 4.15: Ontologie der Entwurfssprache zur Problemdefinition.

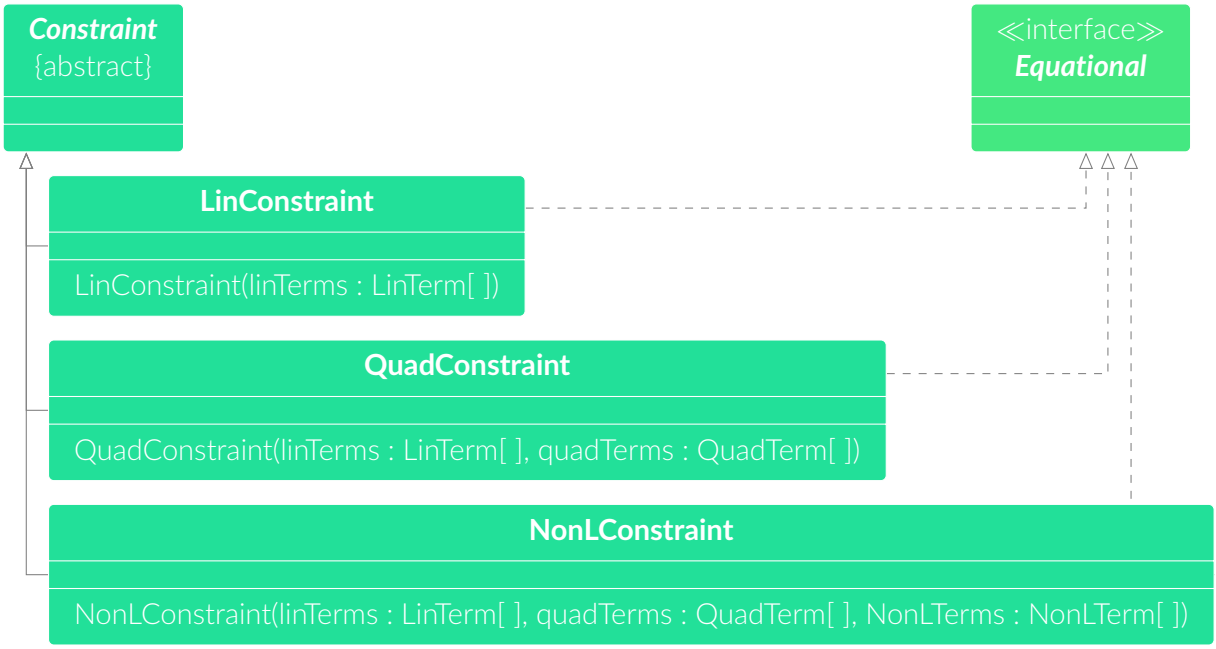


Abbildung 4.16: Ontologie für mathematische Restriktionen und deren Konstruktoren.

durch Übergabe der jeweiligen Terme definiert werden, wobei die Auswertungsmethoden bereits vorhanden sind. Im Allgemeinen kann bei den Zielfunktionen und Restriktionen auch von einer Einteilung in klassische methodenbasierte Programmierung und mathematische Programmierung gesprochen werden. Benötigt wird diese Einteilung, um in der nachfolgenden Problemlösungsphase passende Lösungsstrategien zuordnen zu können. Hierauf wird auf den folgenden Seiten noch gesondert eingegangen.

Es sollen nun die das `Equational` Interface implementierenden Klassen näher betrachtet werden. Das Kürzel `Lin` steht für eine lineare Formulierung. Das bedeutet, dass nur lineare Terme in die Definition der Restriktion oder Zielfunktion eingehen dürfen. Das Kürzel `Quad` steht für eine quadratische Formulierung, in der sowohl quadratische als auch lineare Terme zulässig sind. Das Kürzel `NonL` symbolisiert nichtlineare Restriktionen und kann neben linearen und quadratischen Termen und Terme mit Variablen höheren Grades enthalten sowie eine Sammlung an vordefinierten trigonometrischen Ausdrücken. Diese sind als Enumeration `Expression` in der Ontologie repräsentiert und umfassen zum Zeitpunkt der Verfassung der vorliegenden Arbeit nur die Sinusfunktion (`SIN`) und die Kosinusfunktion (`COS`), welche aber für die Optimierung von Layout- oder Packingproblemen ausreichen. Ein Term ist dabei immer ein Produkt aus Variablen und konstanten Koeffizienten und geht je nach gewähltem Koeffizientenvorzeichen als Summand oder Subtrahend in den Gesamtausdruck ein. Lineare Terme (`LinTerm`) haben die Einschränkung, dass Variablen nur linear eingehen dürfen, quadratische Terme (`QuadTerm`) dürfen eine quadratische Variable oder ein Produkt aus zwei Variablen enthalten und nichtlineare Terme (`NonLTerm`) können beliebig viele Variablen beliebigen Grades und in beliebiger Kombination an trigonometrischen Funktionen beinhalten.

Es bleibt die Frage, wie sichergestellt werden kann, dass Terme ausschließlich mit kompatiblen Graden instanziiert werden können und Restriktionen oder die Zielfunktion nur mit kompatiblen Termen. Dies wird über die definierten Konstruktoren abgesichert. So kann an den Konstruktor der `LinTerm` Klasse nur eine Variable und ein Koeffizient pro Term übergeben werden. An den Konstruktor der `QuadTerm` Klasse müssen genau zwei Variablen und ein Koeffizient übergeben werden, dabei muss es sich zweimal um dieselbe Variable handeln, wenn eine

quadratische Variable gewünscht ist. An den Konstruktor der `NonLTerm` Klasse kann neben dem Koeffizienten eine Liste an Variablen, eine Liste an Hochzahlen dieser Variablen, die deren Grad bestimmen, und eine Liste an Ausdrücken übergeben werden. Die Position in der Liste bestimmt dabei die zugehörigen Paare. Auf dieselbe Weise wird auch die Konsistenz zwischen den speziellen Restriktionen bzw. Zielfunktionen und den Termen sichergestellt. Abb. 4.16 veranschaulicht das beschriebene Schema am Beispiel der mathematischen Restriktionen. Die Konstruktoren sind dabei so aufgebaut, dass bei Instanziierung nur die kompatiblen Terme übergeben werden können.

Es ist zu beachten, dass die Hinzunahme einer Rand- oder Nebenbedingung oder einer Zielfunktion eine Anreicherung des Gesamtproblems mit neuen Variablen, Gleichungen und Ungleichungen oder prozeduralen Beschränkungen bedeutet. Dadurch kann sich auch der Typ des Gesamtproblems ändern. Wenn ein bestimmtes Optimierungsverfahren verwendet werden soll, muss der/die Benutzer/-in bei der Problemdefinition die Eignung des Problems sicherstellen. Der Problemtyp (`ProblemType`) ist dabei von zentraler Bedeutung, denn er gibt Auskunft darüber, welche Optimierungsart für das Problem angewendet werden kann. Aus Gründen der Übersichtlichkeit wird hierfür in Abbildung 4.15 auf Abbildung 4.11 verwiesen, welcher alle Problemtypen zu entnehmen sind. Lineare Probleme werden durch `Linear` oder die Abkürzung `L` gekennzeichnet und zeichnen sich dadurch aus, dass Variablen ausschließlich linear in das Problem eingehen. Die Abkürzung `MI` steht für „mixed integer“ und bedeutet, dass es sich um ein gemischt-ganzzahliges oder auch diskretes Optimierungsproblem handelt. Diese Eigenschaft betrifft potenziell alle Varianten von `MILinear` bis `MINonlinear`. Dabei bedeutet `MI` nicht zwangsläufig, dass ein Problem nicht auch stetig sein kann, die Verfahren werden aber nicht genauer spezifiziert, da alle angebundenen Optimierer auch gemischt-ganzzahlige Probleme lösen können, wenn sie in der Lage sind, quadratische Probleme zu verarbeiten. Ist der dritte Buchstabe der Kennzeichnung ein `Q`, handelt es sich um ein Problem mit einer quadratischen Zielfunktion und linearen Nebenbedingungen. Sind auch quadratische Nebenbedingungen enthalten, wird dies mit der nachfolgenden Buchstabenfolge `QC` gekennzeichnet. Sind Ausdrücke höheren Grades in der Zielfunktion oder den Nebenbedingungen beteiligt oder Trigonometrische Funktionen, wird ein Problem als nichtlinear mit `Nonlinear` gekennzeichnet. Da die Lösbarkeit von Optimierungsproblemen in den meisten Fällen von der Konvexität abhängt, wird auch diese in der Typbezeichnung angegeben. Dabei wird unterschieden, ob ein Problem konvex (`Convex`) oder nicht-konvex (`Nonconvex`) ist (siehe dazu auch *3.4.1 Konvexe oder Nichtkonvexe Probleme*). Viele der angebundenen Optimierer schließen die Lösung von nicht-konvexen Problemen aus. Bei Konvexität muss weiter unterteilt werden in lösbar definit (`SD` für solvable definite) oder unlösbar definit (`ND` für nonsolvable definite). Lösbar definit gilt für Probleme mit konvexen Zielfunktionen, die zu minimieren sind, oder konkaven Zielfunktionen, die zu maximieren sind, sowie mit konvexen Nebenbedingungen mit einer oberen Schranke oder konkaven Nebenbedingungen mit einer unteren Schranke. Auf Konvexität lässt eine positive Semidefinitheit der Hesse-Matrix schließen, auf Konkavität lässt eine negative Semidefinitheit der Hesse-Matrix schließen. Alle anderen Fälle führen zu der Eigenschaft unlösbar definit. Für weitere Ausführungen zu dieser Thematik sei der interessierte Leser auf Abschnitt 3.4.1 verwiesen. Alle Problemtypen bis hier hin beinhalten ausschließlich Neben- und Randbedingungen der Klasse `Equational` und können somit als mathematisches System von (Un-) Gleichungen ausgedrückt werden. Ist das Problem vom Typ `Coded`, bedeutet dies, dass außerdem auch Methoden oder Prozeduren als Nebenbedingungen eingehen. Die angebundenen klassischen globalen Optimierer entfallen dann als Lösungsmöglichkeit und es muss in Konsequenz ein metaheuristisches Verfahren verwendet werden.

Die Namensgebung der Klassen zur Problemdefinition gibt einen Hinweis auf die beteiligten Problemklassen, sodass leicht erkannt werden kann, zu welchem Gesamtproblemtyp die defi-

nierten Nebenbedingungen führen. Darüber hinaus ist in der entwickelten Entwurfssprache ein Packing-Problem so implementiert, dass es selbst erkennt, von welchem Typ es ist. Folglich weiß das Problem selbst, welcher Lösungsalgorithmus oder Solver zu seiner Lösung verwendet werden kann. Dadurch hat die Entwurfssprache die Fähigkeit, dem Benutzer auf Anfrage geeignete Optimierungsverfahren vorzuschlagen, aus denen er dann je nach Verfügbarkeit und Lizenzmodell auswählen kann. Die Abschätzung eines geeigneten Verfahrens basiert dabei auf der Unterscheidung von Rand- und Nebenbedingungen und der Zielfunktion in `Equational` Subklassen und sog. codebasierte Klassen. Im Falle einer `Equational` Subklasse wird außerdem die Definitheit der Hessematrix ausgewertet. Dieses Vorgehen ermöglicht eine einfache Handhabung, da der Nutzer bei der Modellierung des Problems alle Definitionsschritte ohne Kenntnis des genauen mathematischen Charakters der einzelnen Gleichungen durchführen kann. Nach der Problemspezifikation kann der/die Benutzer/-in dann eine Methode aufrufen, die eine Liste aller Optimierungsverfahren ausgibt, die zur Lösung des spezifischen Problemtyps verwendet werden können.

4.3.3 Interdependenz der Problemdarstellung und -lösung

Wie in den vorherigen Kapiteln bereits angesprochen, existieren verschiedene Problemarten bzw. verschiedene Repräsentationsmöglichkeiten und unterschiedliche Optimierungsalgorithmen oder -softwares. Ebenso wurde bereits angesprochen, dass der/die Benutzer/-in bei der Problemdefinition die Eignung des Problems sicherstellen muss, wenn ein bestimmtes Optimierungsverfahren verwendet werden soll. Im Allgemeinen kann davon ausgegangen werden, dass Optimierungsprobleme im Ingenieurwesen immer auch Entscheidungen oder Szenarien berücksichtigen müssen, die sog MIP Probleme (für *mixed integer program*). Für gemischtganzzahlige Problemklasse bildet das lineare Modell den einfachsten Fall und kann von einem sog. MILP Optimierungsverfahren (für *mixed integer linear program*) gelöst werden. Sind Teile des Optimierungsmodells quadratisch, muss bereits ein sog. MIQP Verfahren (für *mixed integer quadratic program*) oder MIQCP Verfahren (für *mixed integer quadratic constrained program*) verwendet werden. Wird die Problemklasse in MIQP oder MIQCP eingestuft, handelt es sich bei der quadratischen Zielfunktion und den quadratischen Nebenbedingungen um die Problemanteile mit höchstem Grad, d.h. es sind durchaus auch lineare Anteile in solchen Problemklassen üblich. Es kann also davon ausgegangen werden, dass sich die Problemklassen gegenseitig erweitern. MIQP und MIQCP erweitern somit MILP Modelle. Optimierungsmethoden, welche MIQP und MICQP Probleme lösen können, müssen in der Lage sein auch die linearen Anteile auszuwerten und können damit auch reine MILP Probleme optimieren. Neben dem Grad der Problemformulierung spielt die Konvexität oder Nichtkonvexität eine entscheidende Rolle für die Wahl eines kompatiblen Lösungsmechanismus. Im Allgemeinen kann davon ausgegangen werden, dass konvexe Problemklassen einfacher zu lösen sind als nichtkonvexe. Das liegt neben dem mathematischen Charakter (der interessierte Leser sei auf *3.4.1 Konvexe oder Nichtkonvexe Probleme* verwiesen) auch daran, dass für nichtkonvexe Probleme deutlich weniger effiziente Algorithmen zur Verfügung stehen, da viele entwickelte Berechnungsmethoden die Konvexität für ihre Lösungsmethode benötigen. In Folge können Optimierungsverfahren für konvexe Probleme nichtkonvexe Probleme auch nicht lösen, andersherum ist es aber durchaus möglich. Da ein nichtkonvexes Problem ein konvexes erweitert, kann ein Optimierungsverfahren für nichtkonvexe Modelle in den meisten Fällen auch konvexe Modelle optimieren, vorausgesetzt der Polynomgrad ist derselbe. Typische nichtkonvexe Problemformulierungen ergeben sich beispielsweise durch die Berücksichtigung von trigonometrischen Funktionen. Diese und höhere Polynomgrade als zwei führen zu den sog. gemischt-ganzzahligen nichtlinearen Optimierungsproblemen (oder kurz MINLP für *mixed integer nonlinear program*). Hierbei ist zu

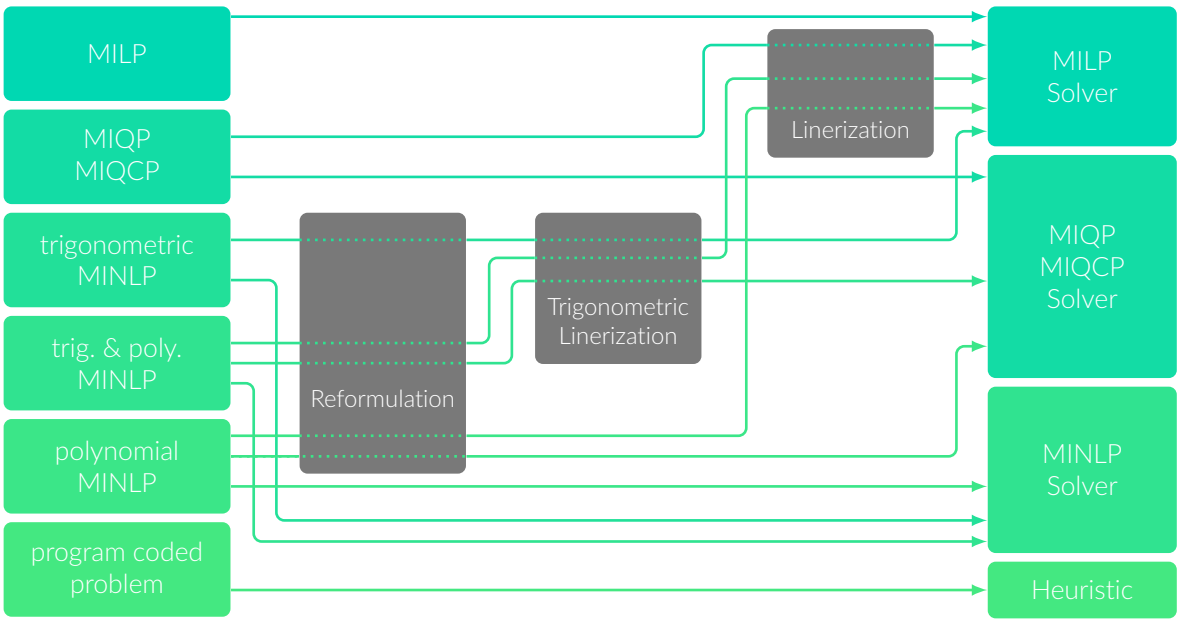


Abbildung 4.17: Abhängigkeit zwischen Problemklassen und Lösungsmethoden mit Reformulierung und Linearisierung zur Vereinfachung der Problemklassen.

beachten, dass quadratische Problemklassen auch als nichtlineare Problemklassen bezeichnet werden können, sie aber an dieser Stelle separat bezeichnet werden, da zahlreiche auf quadratische Ordnungen spezialisierte Lösungsmethoden existieren. Auch hier gilt wieder, das MINLP Problem erweitert das konvexe und nichtkonvexe MIQP oder MICQP Problem.

Zur Modellierung von Packingproblemen müssen in der Regel sowohl nichtlineare Nebenbedingungen als auch ganzzahlige Entscheidungen berücksichtigt werden [Schewe und M. Schmidt, 2019]. Die Nichtlinearität wird dabei vor allem für die Modellierung von rotatorischen Freiheitsgrade benötigt und wird über trigonometrische Funktionen eingebracht. Die Ganzzahligkeit entsteht durch Szenarien, welche in die Modelle eingehen (siehe hierzu *4.3.4 Logische Ausdrücke und Szenarien*), oder sind implizit in den Darstellungen der Kollisionsvermeidung enthalten (siehe dazu *4.4.3 Kollisionsvermeidung*). Solche MINLP Optimierungsprobleme treten häufig spätestens dann auf, wenn die Physik des Packingprozesses im Ingenieurwesen hinreichend genau modelliert werden soll. Leider stehen nicht immer alle Optimierungsverfahren zur Verfügung, um jegliche Problemart adäquat zu lösen. Zwar können „einfachere“ Problemklassen von für „komplexere“ Problemklassen ausgelegte Verfahren gelöst werden, andersherum ist dies aber nicht möglich. Daraus ergibt sich die Notwendigkeit für eine Konvertierung der Problemklassen ineinander. Während der vorliegenden Arbeit sind zwei Verfahren in die vorgestellte Optimierungsentwurfssprache integriert worden, welche auch häufig in der Literatur zu finden sind. Die Linearisierung und die Reformulierung. Diese werden in *4.3.3 Reformulierung* und *4.3.3 Linearisierung* vorgestellt. Verwendung finden sie konkret bei Einsatz der globalen Optimierung, wenn interne Problemrepräsentationen in von Optimierungssoftwares interpretierbare und verarbeitbare Problemformulierungen übersetzt werden müssen (siehe dazu *4.3.4 Mathematische Optimierung als Entwurfssprache*). Zwar reichen diese Methoden leider oft nicht aus, um in die für die Praxis relevanten Größenordnungen von MINLPs vorzustoßen, die zur Lösung realistischer Packingprobleme benötigt werden. Sie können aber für einfache Abschätzungen z.B. mithilfe einfacher Geometrien genutzt werden, um einen globalen Überblick über einen sinnvollen Entwurfsraum zu bekommen oder um eine Vorauswahl an Lösungsvarianten auszuwählen, welche genauer untersucht werden kann.

Im Bereich des Ingenieurwesens fällt es in manchen Fällen leichter, die Problemrepräsentation in Form von Programmcode zu modellieren. Diese Art der Problemdarstellung kann auch als Black-Box Funktion verstanden werden, da Informationen in die jeweiligen Methoden eingehen und Ergebnisse liefern, ohne dass immer genau eingesehen werden kann, was innerhalb des Codes geschieht. Für solche Problemrepräsentationen können keine mathematischen Optimierer genutzt werden, da sie eine Darstellung in Form von mathematischen Funktionen voraussetzen. Es müssen also Optimierungsmethoden eingesetzt werden, welche solche Black-Box-Funktionen auswerten können. Typische Optimierungsmethoden sind hier heuristische stochastische Verfahren wie die in der vorliegenden Arbeit vorgestellte Partikel-Multi-Schwarm-Optimierung. Die Problemrepräsentation spielt hierbei keine große Rolle und es können sowohl mathematische Funktionen sowie Programmcode eingehen. Es sei an dieser Stelle allerdings darauf hingewiesen, dass die Diskussion lediglich die Verarbeitbarkeit der Problemformulierung berücksichtigt und keinerlei Aussagen hinsichtlich Effizienz oder Optimalität gemacht werden sollen. Zwar können alle Problemarten mit heuristischen Verfahren verarbeitet werden, wie sinnvoll dies ist, muss aber im Einzelfall entschieden werden. Ist nun gewünscht, dass mathematische Optimierungsverfahren für in Programmcode repräsentierte Optimierungsprobleme verwendet werden, muss das Optimierungsproblem von Grund auf neu modelliert und durch ein System von mathematischen Funktionen ersetzt werden.

Im Folgenden sollen die beiden zuvor genannten Methoden für die Überführung von Problemklassen ineinander vorgestellt werden.

Reformulierung

Liegt ein gemischt-ganzzahliges nichtlineares Modell vor, kann zur Reduktion der höhergradigen Nichtlinearitäten eine Reformulierung eingesetzt werden, welche zu einer Formulierung minimal zweiten Grades führt. Zu beachten ist hierbei allerdings, dass die Reformulierung ausschließlich für die Anteile des Gesamtproblems eingesetzt werden kann, welche faktorisierbar sind. Für nicht faktorisierbare Nichlinearitäten sei auf die Linearisierung im darauffolgenden Abschnitt verwiesen. Es ergeben sich in der vorliegenden Arbeit zwei Anwendungsfälle für die Reformulierung. Zum einen wird sie genutzt, um Probleme höherer Ordnung auf quadratische Probleme zu reduzieren und zum anderen, um sie so weit aufzuspalten, dass nichtlineare und nicht faktorisierbare Funktionen (für eine spätere Linearisierung) separat vorliegen. Im Folgenden wird die Reformulierung eines beispielhaften Optimierungsproblems vorgenommen (angelehnt an [Linderoth und Luedtke, 2016]):

$$
\begin{aligned}
min \quad & x_1 + x_1 x_2^2 \\
x_1 x_2 + sin(x_2) & \leq 4 \\
-4 \leq x_1 & \leq 4 \\
0 \leq x_2 & \leq 10
\end{aligned}
$$

Es ergibt sich die folgende Reformulierung:

$$
\begin{aligned}
&min \quad x_1 + x_5 \\
&x_4 + x_3 \leq 4 \\
& \\
& \\
&x_3 = sin(x_2) \\
&x_4 = x_1 x_2 \\
&x_5 = x_2 x_4
\end{aligned}
\qquad
\begin{aligned}
& \\
& \\
-4 \leq & x_1 \leq 4 \\
0 \leq & x_2 \leq 10 \\
-1 \leq & x_3 \leq 1 \\
-40 \leq & x_4 \leq 40 \\
-400 \leq & x_5 \leq 400
\end{aligned}
$$

Bei der Reformulierung werden zusätzliche Variablen und Gleichungen eingeführt, um Polynomgrade zu reduzieren. Dabei wird pro reduziertem Grad eine Variable und Gleichung benötigt. Die oberen und unteren Schranken der neuen Variablen können dabei aus den alten Variablen und Restriktionen gewonnen werden. Die Ergebnisse der Reformulierung sind von der Reihenfolge der Ersetzung abhängig, d.h. es gibt immer mehrere korrekte Reformulierungsvarianten. In der vorliegenden Arbeit wird der implementierte Reformulierungsprozess für die Übersetzung des von dem/der Nutzer/-in modellierten Optimierungsproblems in eine softwareabhängige Darstellung verwendet.

Linearisierung

Liegen in einem Problem quadratische Formulierungen vor, kann zunächst kein Lösungsverfahren für lineare Probleme verwendet werden. Um dies aber doch zu ermöglichen, kann eine stückweise lineare Näherung des quadratischen Problems konstruiert werden. Ein weiterer Anwendungsfall für die Linearisierung liegt bei der Verwendung von MIQP oder MICQP Optimierungsverfahren und bei Vorhandensein von nichtlinearen (und nicht faktorisierbaren, denn hierfür kann die Reformulierung genutzt werden) Anteilen vor, wie z.B. trigonometrischen Funktionen. Da diese grundsätzlich nichtlinear sind, kann ein Optimierungsverfahren für lineare und quadratische Probleme nicht mit ihnen umgehen, solange sie nicht linearisiert sind. Die direkte Methode, um stückweise lineare Funktionen zu nutzen, ist eine stückweise lineare Näherung von einer Funktion zu konstruieren und dann statt der originalen Bedingung die stückweise lineare Variante als gemischt-ganzzahliges Optimierungsproblem zu modellieren. Es ist zu beachten, dass die Entscheidung für eine Linearisierung problemabhängig ist und nicht immer zuverlässig funktioniert. In umfangreichen Problemen müssen die stückweise linearen Approximationen eventuell verhältnismäßig grob gewählt werden, um den Löser nicht zu überfordern. Dann kann es aber passieren, dass die Approximationen nicht zusammenpassen und es keine Werte gibt, für die die Gleichungen exakt eingehalten werden. Weiterhin gibt es zahlreiche unterschiedliche Methoden, um eine stückweise Linearisierung als gemischtganzzahliges Problem auszudrücken. Zu dieser Thematik wird auf [Geißler et al., 2012] und [Vielma, 2015] verwiesen. Eine effiziente Variante der Linearisierung ist ein wichtiger Bestandteil für die Lösung von realen Packingproblemen, eine nähere Untersuchung geht jedoch über den Anwendungsbereich der vorliegenden Arbeit hinaus. Viele der zur Verfügung stehenden Optimierungsbibliotheken oder -softwares werden bereits mit Linearisierungsmethoden für elementare Funktionen ausgeliefert. In der vorliegenden Arbeit wird auf die interne Linearisierung der jeweiligen Bibliothek oder Software zugegriffen. Um diese nutzen können, werden Gesamtprobleme mithilfe der zuvor beschriebenen Reformulierung soweit vereinfacht, dass alle Nichtlinearitäten elementar vorliegen.

4.3.4 Mathematische Optimierung als Entwurfssprache

Für die globale mathematische Optimierung von gemischt-ganzzahligen linearen oder nichtlinearen Optimierungsproblemen ist während der vorliegenden Arbeit eine Schnittstelle für die Anbindung von bereits vorhandenen Optimierungssoftwaretools entwickelt und implementiert worden. Zum Zeitpunkt der Verfassung der vorliegenden Arbeit sind Schnittstellen zu insgesamt fünf Open-Source-Lösern verfügbar, die innerhalb der Optimierungsentwurfssprache verwendet werden können. Diese bieten eine alternative Optimierungsressource für Projekte, bei denen keine kommerzielle Lizenz zur Verfügung steht oder welche keine extrem hohe Effizienz erfordern, da sie keine besonders umfangreichen Modelle umfassen. Es sei an dieser Stelle darauf hingewiesen, dass Open-Source-Software unter einer Vielzahl von Lizenzen bereitgestellt wird, die sich in ihren Bestimmungen erheblich unterscheiden, insbesondere was die Weitergabe und kommerzielle Nutzung betrifft. Auf der Website jedes Open-Source-Solvers werden spezifische Lizenzbedingungen angegeben oder verlinkt, die von dem/der Anwender/-in sorgfältig zu prüfen sind. Weiterhin wurden Anbindungen zu drei kommerziellen Optimierungssoftwares entwickelt und implementiert, deren Lizenzen eine akademische Verwendung erlauben. Im Folgenden sollen die angebundenen Softwares kurz vorgestellt werden.

GLPK für Windows *GNU Linear Programming Kit, Version 4.65* 2018 Die GLPK-Bibliothek (GNU Linear Programming Kit) ist für die Lösung umfassender linearer, gemischt-ganzzahliger und anderer verwandter Optimierungsprobleme ausgelegt. Das GLPK-Paket umfasst eine Reihe von Lösungsroutinen, die in Form einer aufrufbaren Bibliothek organisiert sind und unterstützt die GNU MathProg Modellierungssprache, die eine Untermenge der AMPL Sprache ist. GLPK gehört zur Kategorie der freien Software, d.h. die Bibliothek kann unter den von der Free Software Foundation veröffentlichten Bedingungen der GNU General Public License weitergeben sowie modifiziert werden.

LPSolve [Berkelaar et al., 2004] LPSolve ist eine freie Software für die Lösung gemischtganzzahliger linearer Optimierungsprobleme, die auf einer überarbeiteten Simplex-Methode und der Branch-and-Bound-Methode für ganze Zahlen basiert. Die Software löst Modelle, die rein linear sind, und kann mit Variablen umgehen, die binäre oder ganzzahlige Werte annehmen können. Sie kann außerdem auch mit halbkontinuierlichen Variablen und sog. *special ordered sets* (SOS) umgehen. LPSolve steht unter den Bedingungen der GNU lesser general public license (LPGL) zur Verfügung.

Glop von OR-Tools [Perron und Furnon, 2021] Die primäre Software für die Lösung von linearen Optimierungsproblemen im Softwarepaket OR-Tools-Software-Suite ist Glop, der hauseigene Optimierer von Google. Laut eigenen Angaben benötigt er nur geringe Rechenzeiten und ist speichereffizient und numerisch stabil. Die gesamte OR-Tools-Software-Suite und damit auch Glop ist unter den Bedingungen der Apache License 2.0 lizenziert.

CP-SAT von OR-Tools [Perron und Furnon, 2021] CP-SAT ist eine, ebenfalls aus dem Softwarepaket OR-Tools-Software-Suite kommende, Optimierungssoftware, die zur Lösung von ganzzahligen Optimierungsproblemen genutzt werden kann. Zur Lösung der Optimierungsaufgaben werden sog. SAT-Methoden (von engl. „satisfiability“) verwendet. Zur Erhöhung der Rechengeschwindigkeit arbeitet der CP-SAT Löser ausschließlich mit ganzzahligen Variablen

und Konstanten. Das bedeutet, dass alle Optimierungsprobleme ausschließlich mit ganzen Zahlen modelliert werden müssen. Wenn Optimierungsprobleme nicht ganzzahlige Termen enthalten, müssen die zugehörigen Restriktionen mit einem ausreichend großen ganzzahligen Faktor multipliziert werden, sodass sichergestellt ist, dass alle Terme ganzzahlig vorliegen. Wie auch alle anderen Komponenten des OR-Tools-Softwarepakets unterliegt der CP-SAT Löser den Bedingungen der Apache License 2.0.

SCIP Optimization Suite [Gamrath et al., 2020] SCIP ist eine nicht-kommerzielle Optimierungssoftware für gemischt-ganzzahlige lineare und gemischt-ganzzahlige nichtlineare Optimierungsprobleme. Nach eigenen Angaben ist die Software ein Framework, das auf die Bedürfnisse von Experten und Expertinnen der mathematischen Optimierung ausgerichtet ist, die die volle Kontrolle über den Lösungsprozess haben und auf detaillierte Informationen bis hin zum inneren Kern des Optimierers zugreifen wollen [SCIP Optimization Suite, 2022]. In der Standardausführung wird SCIP mit einem umfangreichen Paket verschiedener Plugins zur Lösung von Optimierungsproblemen geliefert und wird unter der ZIB Academic License vertrieben. Als Mitglied einer nicht-kommerziellen oder akademischen Einrichtung kann SCIP zu Forschungszwecken frei verwendet werden.

Couenne von Coin-OR [Lougee, 2003] Couenne ist eine auf einem Branch&Bound-Verfahren basierende Bibliothek zur Lösung von gemischt-ganzzahligen Optimierungsproblemen, bei denen alle Funktionen im Allgemeinen nichtlinear sein können. Der Name der Bibliothek setzt sich aus den Anfangsbuchstaben der Bezeichnung *Convex Over and Under Envelopes for Nonlinear Estimation* zusammem. Ziel der Bibliothek ist die Auffindung von globalen Optima der Optimierungsprobleme. Zur Lösung verwendet Couenne ein Reformulierungsverfahren [Tawarmalani und Sahinidis, 2002] und berechnet so eine lineare Problemapproximation für jedes nichtkonvexe Optimierungsproblem [Belotti et al., 2009]. Die Hauptkomponenten der Bibliothek sind eine Ausdrucksbibliothek, Algorithmen zur Trennung von Linearisationsschnitten für die Reformulierung sowie eine Sammlung an Verzweigungsregeln und Methoden zur Einengung der Schranken für das Branch&Bound-Verfahren. Die Entwicklung von Couenne begann im Jahr 2006 im Rahmen einer Zusammenarbeit zwischen IBM und der Carnegie Mellon University. Es handelt sich um eine Open-Source-Software, welche derzeit unter der Eclipse Public License v1.0 (EPL) veröffentlicht wird. Die EPL ist eine von der Open Source Initiative (OSI) genehmigte Lizenz, Couenne ist damit also OSI-zertifizierte Open-Source-Software.

Mosek [MOSEK ApS, 2019] MOSEK ist ein kommerzielles Softwarepaket für die Lösung von linearen, gemischt-ganzzahligen linearen, quadratischen, gemischt-ganzzahligen quadratischen, quadratisch beschränkten, konischen und konvexen nichtlinearen mathematischen Optimierungsproblemen. Nach eigenen Angaben ist eine Besonderheit der Software der Interior-Point-Optimierer, der auf dem sogenannten homogenen Modell basiert ([E. Andersen und Ye, 1997], [E. Andersen und K. Andersen, 1999], [E. Andersen et al., 2003]). Daneben sind viele der bereits in Abschnitt 3.4.2 angesprochenen typischen Methoden zur Problemlösung integriert. Der Solver wurde von dem dänischen Unternehmen Mosek ApS entwickelt, das 1997 von Erling D. Andersen gegründet wurde.

IBM®ILOG®CPLEX® [IBM ILOG CPLEX, 2019] Das IBM ILOG CPLEX Optimization Studio ist ein kommerzielles Optimierungssoftwarepaket, das gemischt-ganzzahlige Optimierungsprobleme, sehr große lineare Optimierungsprobleme, konvexe und nicht-konvexe quadratische

Programmierprobleme und konvexe quadratisch eingeschränkte Probleme löst. Die Software wurde ursprünglich von Robert E. Bixby entwickelt und ab 1988 von CPLEX Optimization Inc. kommerziell vertrieben. 1997 wurde das Unternehmen von ILOG übernommen und ILOG wiederum wurde 2009 von IBM übernommen. Im Rahmen des Programms *Academic Initiative* stellt IBM Mitgliedern akademischer Einrichtungen das CPLEX Optimization Studio kostenlos zur Verfügung. Die zur Verfügung gestellte Version ist voll funktionsfähig und ohne Einschränkungen hinsichtlich der Modell- oder Suchbaumgröße [IBM Academic Initiative, 2020].

Gurobi™ [Gurobi Optimization, LLC, 2022a] Gurobi ist eine kommerzielle Software zur globalen Optimierung von mathematischen Optimierungsproblemen. Sie wurde im Jahr 2008 von Zonghau Gu, Ed Rothberg und Bob Bixby gegründet. Die Anfangsbuchstaben der Nachnamen der drei Gründer sind die Basis für die Namensgebung der Software. Unterstützt werden lineare Optimierungsprobleme, konvexe und nicht-konvexe quadratische Optimierungsprobleme sowie Optimierungsprobleme mit quadratischen Nebenbedingungen, gemischt-ganzzahlige lineare Optimierungsprobleme, konvexe und nicht-konvexe gemischt-ganzzahlige quadratische Optimierungsprobleme sowie gemischt-ganzzahlige Optimierungsprobleme mit quadratischen Nebenbedingungen und darüber hinaus eine Vielzahl weiterer Problemtypen höheren Grades über interne Linearisierung. Neben den in Abschnitt 3.4.2 genannten Algorithmen werden laut eigener Angaben [Gurobi Optimization, LLC, 2022b] eine nicht-traditionelle Suche und weitere Heuristiken zur Lösung gemischt-ganzzahliger Probleme verwendet. Für die akademische Nutzung steht eine voll funktionsfähige Universitätsversion zur Verfügung. Sie hat keine Beschränkungen hinsichtlich der Modellgröße und kann für den eigenen Gebrauch auf mehreren Rechnern installiert und lizenziert werden.

Das Abstract Factory Entwurfsmuster zur Anbindung von Optimierungssoftwares

Für die Modellierung des Schnittstellenmoduls zur Anbindung von verschiedenen Optimierungssoftwares wird das sog. *Abstract Factory* Entwurfsmuster [Gamma et al., 1996] verwendet. Es eignet sich in diesem Fall besonders gut, da die Schnittstellen zu den Optimierern ohne deren Implementierung in der Ontologie für die Optimierung bereitgestellt werden sollen. Die `AbstractSolverFactory` Klasse definiert dabei ein Interface zur Erzeugung von `AbstractSolver` Objekten. Die konkreten `SolverFactoryXX` Klassen erzeugen die konkreten `SolverXX` Objekte. Die `AbstractSolver` Klasse definiert ein Interface für einen Solvertypen und die `SolverXX` Klasse definiert einen konkreten Solver (bzw. die konkrete Anbindung einer Optimierungssoftware) durch Implementierung des Interfaces und wird durch die korrespondierende konkrete Fabrik erzeugt. Die `MixedIntegerProgramming` Klasse dient als Klient und verwendet die Interfaces der `AbstractSolverFactory` Klasse und der `AbstractSolver` Klasse. Weiterreichende Informationen hinsichtlich des zur Anwendung kommenden Entwurfsmusters sind in [Gamma et al., 1995] zu finden.

Konkret hat diese Umsetzung den Vorteil, dass der/die Nutzer/-in bei Verwendung einer globalen Optimierung über externe Optimierungssoftwares von deren Schnittstellen entkoppelt wird. Er/Sie muss lediglich die gewünschte Optimierungssoftware als Enumerationswert an den Konstruktor der `MixedIntegerProgramming` Klasse angegeben, die Klasse erstellt dann die erforderliche konkrete Fabrik `SolverFactoryXX` selbstständig. Die Fabrik wiederum erzeugt die Anbindung zur zugehörigen Software, hier in Form einer konkreten `SolverXX` Klasse. Soll ein Optimierungsproblem gelöst werden, ruft der/die Anwender/-in die `solve()` Methode der Klasse `MixedIntegerProgramming` auf, welche wiederum die `solveExt(p : Problem)` Methode des konkreten Solvers über das `AbstractSolver` Interface aufruft und dabei das jeweilige Problem übergibt. Jeder konkrete `SolverXX` überschreibt diese Methode und übersetzt darin

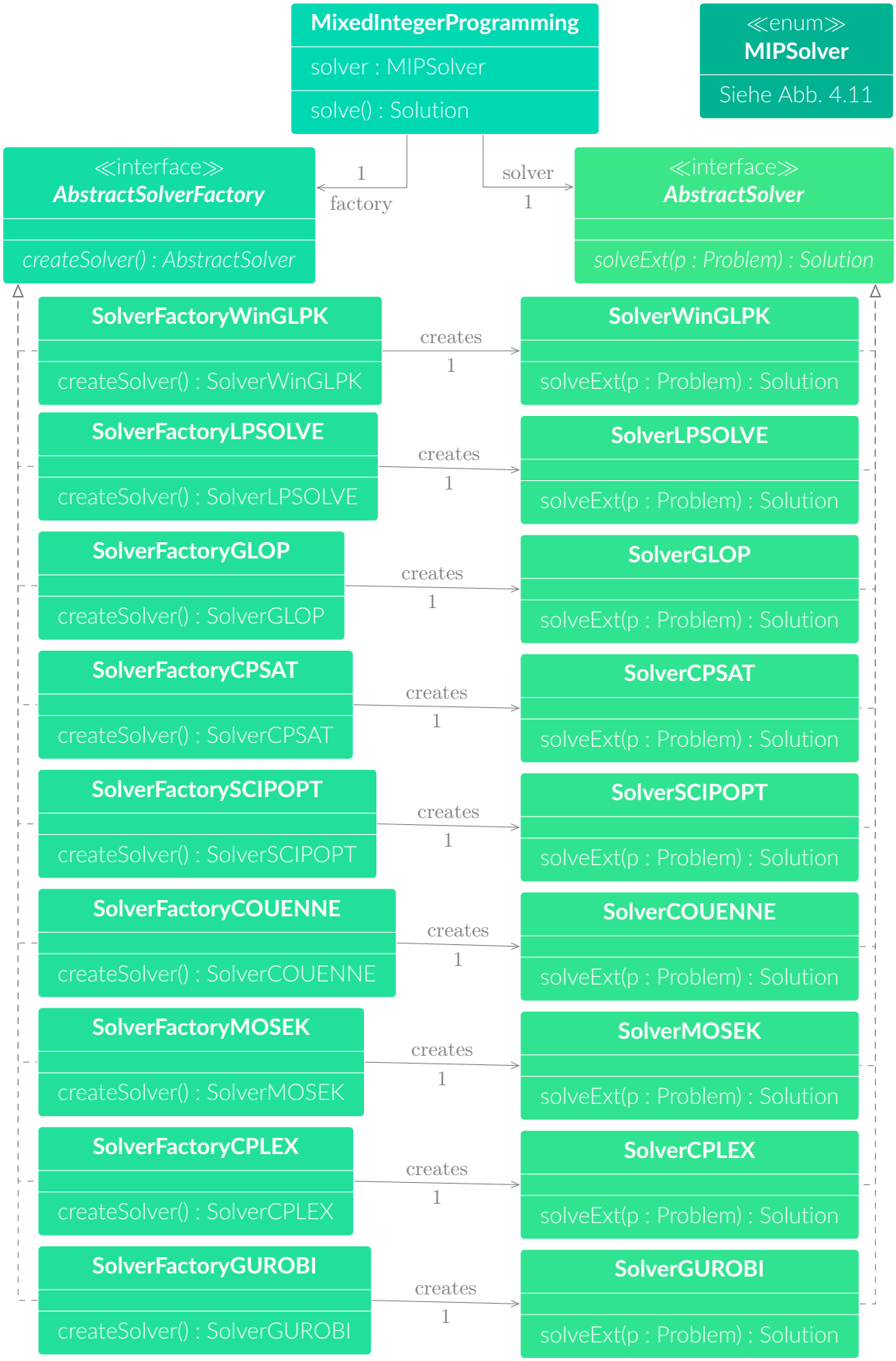


Abbildung 4.18: Ontologie für die Schnittstelle zur Anbindung von Optimierungssoftwares.

das übergebene Optimierungsproblem in eine von der zugehörigen Software interpretierbare und verarbeitbare Problemrepräsentation. Für die Übersetzung werden auch die in *4.3.3 Reformulierung* und *4.3.3 Linearisierung* beschriebenen Ansätze der Reformulierung und/oder Linearisierung verwendet. Ziel ist dabei, den/die Anwender/-in von der Verantwortung für das Erstellen und Konfigurieren der passenden Schnittstelleninstanzen sowie der Übersetzung der Problemdefinition zu entbinden. Die Nutzung der abstrakten Fabrik als Entwurfsmuster hat hier weitere Vorteile. Durch die Entkopplung der `MixedIntegerProgramming` Klasse von einer konkreten Implementierung ist eine hohe Flexibilität gegeben. Außerdem wird ein einfaches Austauschen der zu verwendenden Optimierungssoftware ermöglicht, so kann nach der zentralen Problemdefinition ein Optimierer gegen den anderen getauscht werden, ohne dass eine Änderung notwendig ist. Diese Struktur vereinfacht außerdem die Wartbarkeit und bietet eine unkomplizierte Erweiterbarkeit, da schnell neue Anbindungen an andere Optimierungssoftwares integriert werden können, ohne den bereits bestehenden Code bearbeiten zu müssen.

Logische Ausdrücke und Szenarien

Bei der Modellierung von Packingproblemen spielen logische Zusammenhänge eine wichtige Rolle. Die Modellierung solcher Beziehungen stellt keine allzu große Schwierigkeit dar, solange methaheuristische Verfahren verwendet werden und die logischen Abfragen in Programmcode abgebildet werden können. Muss die Modellierung allerdings über Systeme von mathematischen Funktionen stattfinden, stellt sie den/die Anwender-/in vor eine größere Herausforderung. Solche Zusammenhänge müssen dann aussagenlogisch modelliert werden und anschließend in äquivalente mathematische Ungleichungen umgeformt werden. Dabei wird für jede einzelne logische Aussage (wahr oder falsch) eine entsprechende boolesche Variable eingeführt. Bei komplexen Formeln kann es trotzdem mitunter schwierig sein, eine äquivalente mathematische Darstellung zu finden. Zur Unterstützung der Modellierung gibt es genaue Regeln, welche mathematischen Darstellungen bestimmte logische Verknüpfungen ersetzen. Weiterhin existiert eine Methodik, die eine beliebige Formelmenge in eine mathematische Darstellung überführt. Diese besteht aus der Transformation in die sog. Konjunktive Normalform und deren Transformation in die mathematische Darstellung. Der Methodik liegt die Annahme zugrunde, dass sich jede aussagenlogische Formel in eine semantisch äquivalente Formel transformieren lässt. Die einzelnen in der Methodiken enthaltenen Schritte lassen sich der Literatur entnehmen, der interessierte Leser sei an dieser Stelle auf [Suhl und Mellouli, 2007] verwiesen. Um dies zu vereinfachen, stellen einige Optimierungssoftwares sog. *indicator constraints* zur Verfügung, welche logische Beziehungen an boolesche Variablen koppeln. Dabei muss der/die Anwender/-in für die logischen Szenarien boolesche Variablen definieren und festlegen, für welchen Wert welche Zustände auftreten. Typischerweise können zur Modellierung die folgenden logischen Aussagen verwendet werden, hier am Beispiel der Zustände z_1 und z_2:

- Negation: $\neg z$
- Konjunktion: $z_1 \wedge z_2$
- Disjunktion: $z_1 \vee z_2$
- Logische Implikation: $z_1 \rightarrow z_2$ bzw. $\neg z_1 \vee z_2$
- Logische Äquivalenz: $z_1 \leftrightarrow z_2$ bzw. $(\neg z_1 \vee z_2) \wedge (z_1 \vee \neg z_2)$
- Exklusives oder: $z_1 \bigotimes z_2$ bzw. $(z_1 \wedge \neg z_2) \vee (\neg z_1 \wedge z_2)$

Die Modelle der im nachfolgenden Kapitel vorgestellten Packing-Entwurfssprache enthalten Implikationen, setzen aber nicht voraus, dass Indikatorrestriktionen zur Verfügung stehen. Daher sind die in den Modellen enthaltenen mathematischen Gleichungs- und Ungleichungssysteme wie oben beschrieben umformuliert. Die Packingmodelle selbst werden in Abschnitt 4.4 behandelt. Die Umformulierung einer Implikation dagegen soll an dieser Stelle aufgezeigt werden. Um diese zu modellieren, wird in der Literatur oftmals die sog. *Big M* Methode verwendet. Betrachten wir einen einfachen Anwendungsfall mit zwei möglichen Zuständen:

$$x \leq const_a \text{ wenn Zustand } a \tag{4.61}$$
$$x \geq const_b \text{ wenn Zustand } b \tag{4.62}$$

Während Ungleichungen direkt übernommen können, muss eine Gleichung für dieses Verfahren in ein System mit zwei Ungleichungen umformuliert werden.

$$x = const_a \text{ wenn Zustand } a \;\Rightarrow\; \begin{cases} x \leq const_a & \text{wenn Zustand } a \\ x \geq const_a & \text{wenn Zustand } a \end{cases} \tag{4.63}$$

Um dies als Gesamtproblem auszudrücken, müssen sog. Indikatorvariablen Λ_a und Λ_b eingeführt werden. Dann können die Gleichungen angeglichen werden, wie im Folgenden dargestellt.

$$\Lambda_a = \begin{cases} 1 & \text{wenn Zustand } a \text{ eintritt,} \\ 0 & \text{undefiniert} \end{cases} \tag{4.64}$$

$$\Lambda_b = \begin{cases} 1 & \text{wenn Zustand } b \text{ eintritt,} \\ 0 & \text{undefiniert} \end{cases} \tag{4.65}$$

$$\Lambda_a + \Lambda_b = 1 \tag{4.66}$$

$$x \leq const_a + A \cdot (1 - \Lambda_a) \tag{4.67}$$

$$x \geq const_b - B \cdot (1 - \Lambda_b) \tag{4.68}$$

Die Terme A und B sind Konstanten, welche ausreichend groß sein müssen, sodass die Ungleichungen bei $\Lambda_a = 0$ oder $\Lambda_b = 0$ für jeden möglichen Variablenwert von x gelten. Diese Konstanten werden in der Literatur häufig mit M bezeichnet, woraus sich der Name der *Big M* Methode ableitet. Die Zustandssummengleichung Gl. (4.66) stellt dabei sicher, dass genau einer der beiden Zustände eintritt. Es ist zu beachten, dass es sich bei der Formulierung von zustandsabhängigen Restriktionen mithilfe der Big M Methode um Implikationen handelt. Es gilt also am Beispiel von Gleichung (4.67):

$\Lambda_a = 1 \quad \rightarrow \quad x < const_a$

d.h. $\Lambda_a = 1$ ist eine hinreichende Bedingung für $x < const_a$

oder $x < const_a$ ist eine notwendige Bedingung für $\Lambda_a = 1$.

Man erhält aber aus $\Lambda_a \neq 1$ und somit $\Lambda_a = 0$ keine Aussage über die Aktivität der Restriktion $x < const_a$, es gilt ja stattdessen $x < const_a + A$, eine Formel, welche immer erfüllt ist. Formel

$x < const_a$ kann wahr sein, genauso wie sie falsch sein kann. „Ex falso sequitur quodlibet“ - „Aus falschem folgt Beliebiges“. Daher wird im Folgenden von einer *Aktivierung* oder *Deaktivierung* der Restriktionsgleichungen gesprochen. Ist die zugehörige boolesche Variable wahr bzw. gleich eins, schränkt die Gleichung den Zustandsraum ein und ist damit *aktiv*. Ist die zugehörige boolesche Variable falsch bzw. gleich null, ist die Gleichung für alle sonstigen Variablenwerte immer wahr und wirkt nicht mehr restriktiv auf den Zustandsraum. Dann wird sie als *inaktiv* bezeichnet. Das ganze kann auch mithilfe der Aussagenlogik ausgedrückt werden. Es gelten die folgenden Beziehungen:

$$\Lambda_a \rightarrow x \leq const_a : \quad \neg\Lambda_a \vee x \leq const_a \tag{4.69}$$
$$\Lambda_b \rightarrow x \geq const_b : \quad \neg\Lambda_b \vee x \geq const_b \tag{4.70}$$

Durch die Zustandssummengleichung sind nur zwei mögliche Kombinationen der Indikatorvariablen möglich, woraus sich die folgende Wahrheitstafel ergibt:

Λ_a	Λ_b	
w	f	valide
f	w	
w	w	invalide n. Gl. (4.67)
f	f	

Tabelle 4.1: Wahrheitstafel für Indikatorvariablen.

Es sollen nun nur noch die beiden oberen validen Kombinationen der Indikatorvariablenwerte betrachtet werden. Damit sind noch die folgenden Aussagen möglich:

Λ_a	$x \leq const_a$	
f	w	valide
f	f	
w	w	
w	f	invalide n. Gl. (4.69)

Tabelle 4.2: Wahrheitstafel Zustand a.

Λ_b	$x \leq const_b$	
w	f	invalide n. Gl. (4.70)
w	w	valide
f	f	
f	w	

Tabelle 4.3: Wahrheitstafel Zustand b.

Für einen konkreten Anwendungsfall soll ein Beispiel betrachtet werden, welches dem Leser in Abschnitt 4.2.2 bereits in ähnlicher Weise begegnet ist. In diesem sollen zwei rechteckige Objekte in zwei Dimensionen relativ zueinander positioniert werden. Insgesamt können die relativen Positionen mit vier Variablen, einer Zustandskombinationsgleichung und vier Positionierungsgleichungen repräsentiert werden:

$$X^{-}_{i,j} = \begin{cases} 1 & \text{das Objekt } i \text{ befindet sich links des Objektes } j, \\ 0 & \text{undefiniert} \end{cases} \tag{4.71}$$

$$X_{i,j}^{+} = \begin{cases} 1 & \text{das Objekt } i \text{ befindet sich rechts des Objektes } j, \\ 0 & \text{undefiniert} \end{cases} \tag{4.72}$$

$$Y_{i,j}^{-} = \begin{cases} 1 & \text{das Objekt } i \text{ befindet sich vor dem Objekt } j, \\ 0 & \text{undefiniert} \end{cases} \tag{4.73}$$

$$Y_{i,j}^{+} = \begin{cases} 1 & \text{das Objekt } i \text{ befindet sich hinter dem Objekt } j, \\ 0 & \text{undefiniert} \end{cases} \tag{4.74}$$

$$X_{i,j}^{-} + X_{i,j}^{+} + Y_{i,j}^{-} + Y_{i,j}^{+} = 1 \tag{4.75}$$

$$x_i + l_i \leq x_j + L \cdot (1 - X_{i,j}^{-}) \tag{4.76}$$

$$x_j + l_j \leq x_i + L \cdot (1 - X_{i,j}^{+}) \tag{4.77}$$

$$y_i + w_i \leq y_j + W \cdot (1 - Y_{i,j}^{-}) \tag{4.78}$$

$$y_j + w_j \leq y_i + W \cdot (1 - Y_{i,j}^{+}) \tag{4.79}$$

Insgesamt gibt es in diesem Problem vier mögliche aktive Zustände. Jeder mögliche Zustand wird durch eine eigene boolesche Variable repräsentiert, welch zu eins wird, wenn der Zustand wahr ist und zu null, wenn darüber keine Aussage getroffen werden kann. Zu Sicherstellung, dass nur valide Zustandskombinationen erlaubt sind, muss außerdem notwendigerweise eine die Zustandsvariablen aufsummierende Zusatzbedingung formuliert werden. Diese stellt zum einen sicher, dass mindestens eine Restriktion des Zustandsraumes aktiv ist, zum anderen verhindert sie auch das gleichzeitige Eintreten von sich gegenseitig ausschließenden Zuständen. Hier sei nochmal auf den implikativen Charakter der Formulierungen hingewiesen. Ist die Zustandsvariable $X_{i,j}^{-}$ gleich eins, müssen laut Gl. (4.75) alle anderen booleschen Variablen gleich null sein. Tritt dieser Fall ein, bedeutet dies nicht, dass die anderen Zustände nicht eintreten können, es bedeutet lediglich, dass die zugehörigen Zustandsgleichungen inaktiv sind und damit keine weitere Aussage über die Zustände getroffen werden kann. Im konkreten Beispiel würde sich das Objekt i links des Objektes j befinden, gleichzeitig könnte es auch ober- oder unterhalb liegen. Da sich die Gleichungen Gl. (4.76) und Gl. (4.77) aber widersprechen, kann das Objekt i nicht gleichzeitig auch rechts des Objektes j liegen.

Obwohl die vorgestellte Methode ein einfaches und flexibles Instrument für die Modellierung von Implikationen bietet, ist in der Literatur bekannt, dass sie Nachteile bei der Relaxation während des Lösungsprozesses hat und zu geringer numerischer Genauigkeit führen kann. Der interessierte Leser sei hierzu auf [Bai und Rubin, 2009] verwiesen. Da sie jedoch von allen Optimierern interpretiert und verarbeitet werden kann, ist sie die erste Wahl für die Modellierung von Zuständen in der vorliegenden Arbeit. Der Vollständigkeit halber soll an dieser Stelle erwähnt werden, dass in der Literatur eine Vielzahl weiterer Methoden existiert. Weiterreichende Informationen hierzu finden sich in [Bai und Rubin, 2009], [Codato und Fischetti, 2006] und [Belotti et al., 2016].

Die Berechnung von Big M im Framework Die Wahl der Big M Konstante bei der Modellierung von Zuständen muss wohlüberlegt sein. Dabei muss sie möglichst klein gewählt werden, um mögliche numerische Ungenauigkeiten zu umgehen. Gleichzeitig muss sie aber groß genug sein, um die zuverlässige Deaktivierung von Restriktionen sicherzustellen. In der vorliegenden

Arbeiten wurde eine Klasse entwickelt, die zu einer gegebenen Restriktion eine passende Big M Konstante errechnet. Die Klasse `BigM` bietet dafür die `getBigM` Methode, an die die betroffene Ungleichung übergeben wird. Die gesuchte Konstante ist genau dann hinreichend klein und notwendig groß gewählt, wenn sie bei Deaktivierung der Ungleichung genau den Grenzfall für die Ungleichung abbildet, die Ungleichung also zur Gleichung wird. Bei der Berechnung der Konstante wird also die Ungleichung zunächst als Gleichung betrachtet und alle Terme sowie die Konstante auf die linke Seite gebracht. Die gesuchte Konstante ist dann das Minimum der linken Seite für Zwänge mit einer unteren Schranke und das Maximum der linken Seite für Zwänge mit einer oberen Schranke.

$$\sum_{i=0}^{I} term_i \geq const - M \cdot (1 - \Lambda) \tag{4.80}$$

$$\xRightarrow{Grenzfall,\ \Lambda=0} min\Big(-\sum_{i=0}^{I} term_i\Big) - const = M \tag{4.81}$$

$$\sum_{i=0}^{I} term_i \leq const + M \cdot (1 - \Lambda) \tag{4.82}$$

$$\xRightarrow{Grenzfall,\ \Lambda=0} max\Big(\sum_{i=0}^{I} term_i\Big) - const = M \tag{4.83}$$

Es ist zu beachten, dass die Konstante nur sinnvoll zum Einsatz kommt, wenn sie positiv ist. Sollte die Berechnung der Konstante zu einem negativen Ergebnis führen, muss die Restriktion nicht aktiviert (durch lokale Validität) oder inaktiviert (durch globale Aktivität) werden, da sie sowieso für jeden möglichen Variablenwert gilt und dadurch keine einschränkende Wirkung auf den Entwurfsraum hat. Die gesamte Gleichung kann dann im Gesamtproblem vollständig vernachlässigt werden.

Effiziente Modellierung von Szenarien In den in der vorliegenden Arbeit bisher betrachteten Fällen wurden Szenarien durch boolesche Zustandsvariablen dargestellt, welche bei Eintreten eines Szenarios zu eins wurden und zu null sonst. Wie in Abschnitt 4.3.4 erläutert, wirkt sich dabei nur ein Wert der Variable (=1) restriktiv auf die Eingrenzung der Szenarien aus. Daher werden sie im Folgenden als unvollständig-restriktive Variablen bezeichnet. Dadurch ergeben sich mögliche Variablenwertkombinationen, welche keinerlei einschränkende Wirkung auf den Zustandsraum haben, oder sogar invalide sind. Um dies zu umgehen, müssen zur weiteren Restriktion zusätzliche Gleichungen in das Problem mit eingehen, die Zustandssummengleichungen, welche das Eintreten von ausschließlich validen Zustandskombinationen sicherstellen. Eine kompaktere Problemformulierung kann nun erreicht werden, wenn nicht nur einer, sondern beide Variablenwerte restriktiv auf den Zustandsraum wirken. Dadurch lässt sich die Anzahl der notwendigen Variablen für die zu repräsentierenden Zustände minimieren und potenziell sogar die Zusatzgleichung überflüssig machen. Es gilt jedoch eine Einschränkung: Die beiden Zustände, welche von einer Variablen repräsentiert werden, sind nie gleichzeitig aktiv und wirken daher nie gleichzeitig restriktiv auf den Zustandsraum. Diese Voraussetzung ist zwar in vielen Modellen gegeben, muss jedoch für den Einzelfall geprüft werden. Eine boolesche Variable, deren Werte beide restriktiv auf den Zustandsraum wirken, wird im Folgenden als vollständig-restriktive Variable bezeichnet. An dieser Stelle soll das Beispiel aus Abschnitt 4.3.4 wiederverwendet werden. Es gilt dann das folgende kompaktere Ersatzproblem:

$$\Theta^{ab} = \begin{cases} 1 & \text{wenn Zustand } a \text{ eintritt } (\Lambda_a = 1), \\ 0 & \text{wenn Zustand } b \text{ eintritt } (\Lambda_b = 1) \end{cases} \tag{4.84}$$

$$x < const + M \cdot (1 - \Theta^{ab}) \tag{4.85}$$

$$x > const - M \cdot (\Theta^{ab}) \tag{4.86}$$

Die größte Einsparung wird erzielt, wenn die Anzahl der möglichen Zustände gleich einer ganzzahligen Potenz zur Basis zwei ist. In diesen Fällen ist jegliche Kombination der booleschen Variablenwerte valide und es ist keine zusätzliche restriktive Gleichung notwendig. Es gilt also, dass im *klassischen* Fall die Anzahl der repräsentierbaren Zustände der Anzahl boolscher Variablen entspricht und im *kompakten* Fall die Anzahl der repräsentierbaren Zustände 2 hoch der Anzahl boolscher Variablen entspricht, wobei die Bezeichnung *klassisch* für die bereits betrachtete Modellierung steht und die Bezeichnung *kompakt* für die nun neu eingeführte Variante mithilfe der vollständig-restriktiven Variablen.

$$Klassisch : n_{States} = n_{\Lambda} \tag{4.87}$$

$$Kompakt : n_{States} = 2^{n_{\Theta}} \;\Rightarrow\; n_{\Theta} = log_2(n_{States}) \tag{4.88}$$

Werden vollständig-restriktive Zustandsvariablen verwendet, müssen außerdem die Big M Formulierungen der Gleichungen angepasst werden. Die Ausdrücke in Klammern zur Multiplikation mit der Big-M Konstante ergeben sich dabei aus der Nummer des Zustandes in Binärform. Dabei gilt für eine im Zustand s aktive Gleichung immer mit σ als Wert 0 oder 1 an der Stelle t in der Binärzahl der Zustandsnummer von s:

$$Gleichung_s < const + M \cdot \sum_{t=0}^{t=n_{\Theta}-1} \left(\sigma_t + (-1)^{\sigma_t} \cdot \Theta_t\right) \qquad s = 0, ..., n_{States} - 1 \tag{4.89}$$

Ist die Anzahl der zu repräsentierender Zustände ungleich einer ganzzahligen Potenz zur Basis zwei, kann entweder die nächstkleinere ganzzahlige Potenz zur Basis zwei an Zuständen durch vollständig-restriktive Variablen repräsentiert werden und die restlichen benötigten Zustände durch unvollständig-restriktive Variablen oder die nächstgrößere ganzzahlige Potenz zur Basis zwei an Zuständen wird durch vollständig-restriktive Variablen repräsentiert und es müssen die invaliden Zustände bzw. Variablenwertkombinationen durch zusätzliche restriktive Gleichungen verboten werden. Bei der Modellierung muss außerdem berücksichtigt werden, welche Zustände sich gegenseitig ausschließen. Ein gegenseitiger Ausschluss kann leicht erzielt werden, wenn die jeweiligen Zustände durch eine gemeinsame boolesche Zustandsvariable repräsentiert werden, da immer nur einer der beiden möglichen Werte der Variable wahr sein kann. In den folgenden Kapiteln werden zahlreiche mathematische Formulierungen für die im Laufe der Arbeit entwickelten und implementierten Nebenbedingungen vorgestellt. Der Leser sei an dieser Stelle darauf hingewiesen, dass diese soweit möglich in der erläuterten Form möglichst effizient gestaltet sind, allerdings nicht auf jeden Einzelfall gesondert eingegangen wird. Abschließend soll der interessierte Leser auf [Vielma und Nemhauser, 2008] und [Vielma und Nemhauser,

2011] verwiesen werden, in deren Veröffentlichungen vergleichbare Vereinfachungen mit binären Variablen diskutiert werden.

Zur Verdeutlichung soll im Folgenden das bereits in Abschnitt 4.3.4 eingeführte konkrete Beispiel betrachtet werden, in dem zwei rechteckige Objekte in zwei Dimensionen relativ zueinander positioniert werden. Wird Gl. (4.88) zu Grunde gelegt, können vier Zustände auch durch zwei boolesche Variablen dargestellt werden, ohne dass eine zusätzliche Gleichung benötigt wird. Dadurch kann für das zweidimensionale Positionierungsproblem von zwei rechteckigen Objekten eine effizientere Formulierung mit lediglich zwei booleschen Variablen erzielt werden:

$$\Theta_{i,j}^{xy} = \begin{cases} 1 & \text{wenn eine Verschiebung des Objektes } i \text{ zum Objekt } j \text{ in} \\ & \textbf{Y-Koordinatenrichtung} \text{ vorliegt,} \\ 0 & \text{wenn eine Verschiebung des Objektes } i \text{ zum Objekt } j \text{ in} \\ & \textbf{X-Koordinatenrichtung} \text{ vorliegt} \end{cases} \tag{4.90}$$

$$\Theta_{i,j}^{+/-} = \begin{cases} 1 & \text{wenn die Verschiebung des Objektes } i \text{ zum Objekt } j \textbf{ positiv} \text{ ist,} \\ 0 & \text{wenn die Verschiebung des Objektes } i \text{ zum Objekt } j \textbf{ negativ} \text{ ist} \end{cases} \tag{4.91}$$

$$x_i + l_i \leq x_j + L \cdot (\Theta_{i,j}^{xy} + \Theta_{i,j}^{+/-}) \tag{4.92}$$

$$x_j + l_j \leq x_i + L \cdot (1 + \Theta_{i,j}^{xy} - \Theta_{i,j}^{+/-}) \tag{4.93}$$

$$y_i + w_i \leq y_j + W \cdot (1 - \Theta_{i,j}^{xy} + \Theta_{i,j}^{+/-}) \tag{4.94}$$

$$y_j + w_j \leq y_i + W \cdot (2 - \Theta_{i,j}^{xy} - \Theta_{i,j}^{+/-}) \tag{4.95}$$

Das ganze kann verdeutlicht werden, wenn alle möglichen Zustände betrachtet werden, die durch die beiden booleschen Variablen repräsentiert sind:

$$\Theta_{i,j}^{xy}\ \Theta_{i,j}^{+/-} = \begin{cases} 0\,0 & \text{wenn } X_{i,j}^{-} = 1 \text{ (Objekt } i \text{ links von Objekt } j) \\ 0\,1 & \text{wenn } X_{i,j}^{+} = 1 \text{ (Objekt } i \text{ recht von Objekt } j) \\ 1\,0 & \text{wenn } Y_{i,j}^{-} = 1 \text{ (Objekt } i \text{ vor Objekt } j) \\ 1\,1 & \text{wenn } Y_{i,j}^{+} = 1 \text{ (Objekt } i \text{ hinter Objekt } j) \end{cases} \tag{4.96}$$

Hieraus wird ersichtlich, dass ausschließlich die validen Zustände durch Variablen repräsentiert sind, und es somit keiner weiteren Einschränkung durch zusätzliche Zusatzbedingungen bedarf. Die beschriebene Problemformulierung kommt in der vorliegenden Arbeit im Anwendungsbeispiel aus Abschnitt 5.1.2 zum Einsatz.

4.3.5 Partikel-Multi-Schwarm Optimierung als Entwurfssprache

Bei hochdimensionalen Packingproblemen sind deterministische mathematische Optimierungsalgorithmen häufig nicht in der Lage, in einer annehmbaren Rechenzeit ein zufriedenstellendes Ergebnis zu berechnen. Das liegt unter anderem daran, dass Packingprobleme NP-schwer sind und in der Regel der Suchraum in Relation zur Problemgröße exponentiell ansteigt, was sich direkt auf das Wachstum des Rechenaufwands übertragen lässt. Eine Alternative zur deterministischen mathematischen Optimierung bieten stochastische Verfahren, welche oft in

der Lage sind, einen besseren Kompromiss zwischen Lösungsqualität und Rechenzeit zu bieten. Da wären beispielsweise die heuristischen Algorithmen. Diese sind allerdings substanziell problemabhängig und daher nicht geeignet für ein allgemeines Packingframework. Eine problemunabhängige, aber dennoch stochastische Variante ist die Metaheuristik. Diese kann als ein algorithmisches problemunabhängiges high-level Framework bereitgestellt werden, welches Strategien und Hilfsmittel für die konkrete Problemlösung enthält. Die Problemanpassung an das konkrete Problem findet dabei durch den/die Anwender/-in über ein Finetuning der intrinsischen Parameter der Metaheuristik statt. Damit scheint die Metaheuristik ein vielversprechender Ansatz für die Optimierung von Packingproblemen zu sein.

In diesem Sinne ist in der vorliegenden Arbeit als metaheuristische Bibliothek eine Partikel-Multi-Schwarm-Optimierung (PMSO) implementiert worden. Die Wahl ist auf die PMSO gefallen, da sie flexibel und einfach in der Anwendung ist und auch große Entwurfsräume in akzeptabler Zeit durchsuchen kann. Insgesamt gibt es eine moderate Anzahl an manipulierbaren Parametern, die auch bei gleichen Werten für eine Vielzahl von unterschiedlichen Aufgaben funktionieren. Ein weiteres Kriterium für die Wahl einer PMSO ist, dass sie ohne großen Zusatzaufwand parallelisiert werden kann.

Die Funktionsweise einer klassischen Partikel-Schwarm-Optimierung (PSO) orientiert sich an seinem natürlichen Vorbild, dem biologischen Schwarmverhalten. Die Grundidee ist, dass ein Schwarm aus einer Population von Individuen besteht, die so lange durch einen Suchraum bewegt werden, bis eine Lösung gefunden wird. Die Position eines Individuums stellt dabei eine potenzielle Lösung dar und wird in jedem Zeitschritt neu berechnet. Wie auch in der Natur, in der Gruppenmitglieder versuchen, in der Nähe ihrer Gruppe zu bleiben, orientieren sich die Individuen hinsichtlich ihrer eigenen Position an den Positionen der anderen Mitglieder des Schwarms. Dadurch entsteht eine Kooperation der einzelnen Individuen miteinander, welche schließlich zu einem zum Optimum konvergierenden Verhalten des Verfahrens führen soll. Die Partikel-Multi-Schwarm-Optimierung (PMSO) ist eine Erweiterung oder Variation der klassischen PSO, in der zusätzlich zu einer Reihe an Individuen auch eine Reihe an Schwärmen betrachtet wird, welche auf der Suche nach einer optimalen Lösung zusammenwirken. Es kann davon ausgegangen werden, dass die Suchfähigkeit und Leistung der PMSO die der PSO übersteigen. Diese Erkenntnis ist durch vergleichende Untersuchungen bereits in der Literatur verifiziert worden [H. Zhang, 2011].

Die Ontologie der integrierten Partikel-Multi-Schwarm-Optimierung ist in Abb. 4.19 dargestellt. Die Klasse `ParticleMultiSwarmOptimization` ist dem Leser bereits in Abb. 4.11 begegnet und besteht wie beschrieben aus einem Multischwarm (`MultiSwarm`) und dessen Bestandteilen. Zunächst gehören dazu mehrere eigenständige Schwärme (`Swarm`), deren Anzahl von dem/der Anwender/-in bei Instanziierung vorgegeben werden kann. Zu jedem Schwarm gehört eine ebenfalls vorab spezifizierte Anzahl an einzelnen Individuen, den sog. Partikeln (`Particle`). Jedes der drei genannten Elemente ist in der Lage sein persönliches bestes gefundenes Ergebnis (`bestState`) in Form von Koordinaten abzuspeichern. Die konkrete Umsetzung erfolgt hier über eine Map, eine Datenstruktur für die paarweise Zuordnung von Variablen zu bestimmten Werten. Diese paarweise Zuordnung soll im Folgenden als Position bezeichnet werden. Die beste Position des Partikels wird von ihm selbst entdeckt, die besten Positionen des Schwarms und des Multischwarms ergeben sich aus deren Mitgliedern und sind letztendlich auf die besten Ergebnisse der zugehörigen Partikel zurückzuführen. Mithilfe einer gegebenen Position kann die Zielfunktion am entsprechenden Punkt des Entwurfsraums ausgewertet werden sowie errechnet werden, inwieweit die spezifizierten Rand- und Nebenbedingungen eingehalten werden. Neben der besten Positionen werden so auch die passende Zielfunktionsauswertung (`bestStateFitness`) und die Restriktionsverletzung (`bestStateViolation`) mit

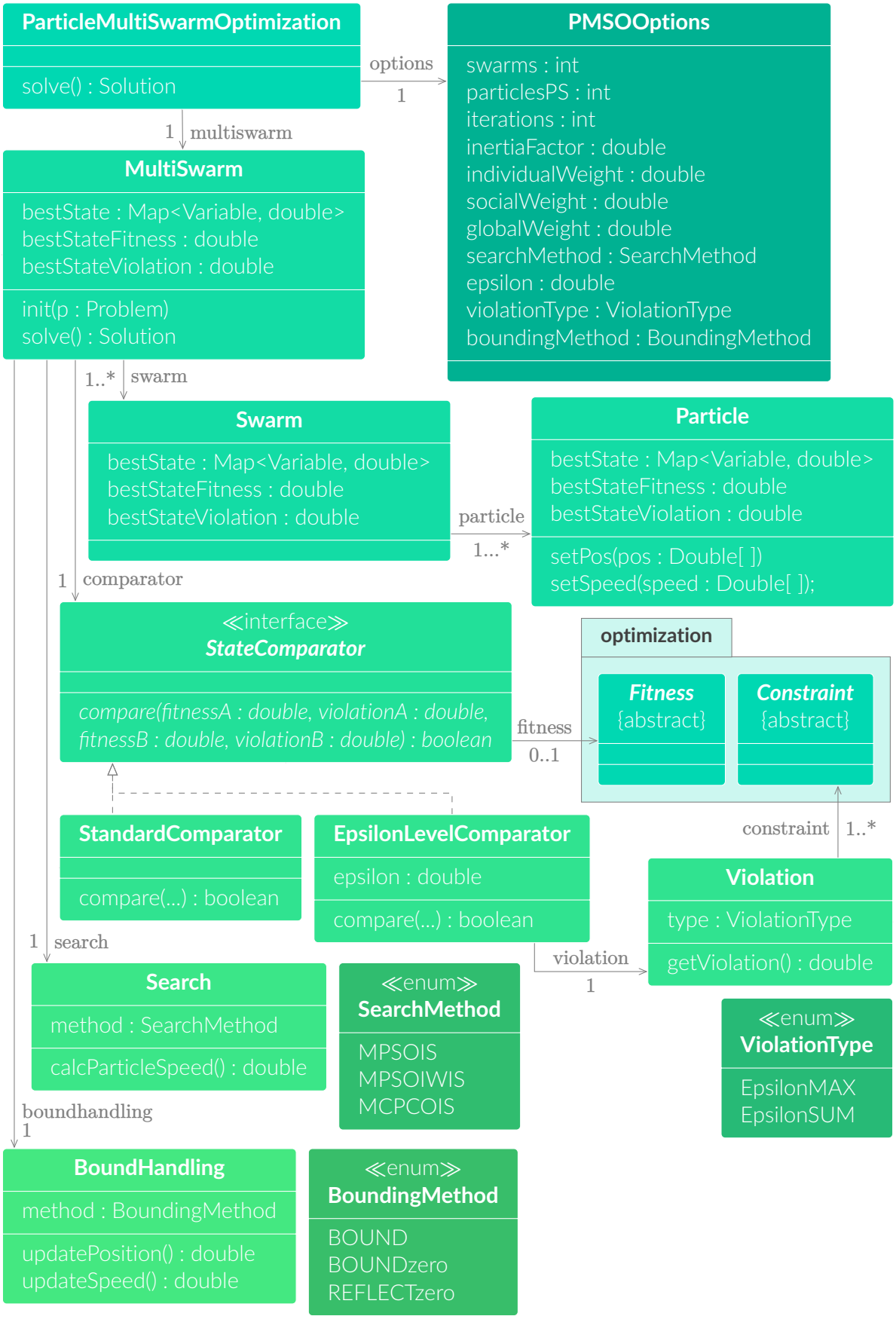


Abbildung 4.19: Ontologie der Partikel-Multi-Schwarm-Optimierung (PMSO)

abgespeichert. In diesem Kontext stellt sich die Frage nach der Bewertungstaktik, mit welcher eine Position bewertet und eingestuft wird. Besteht eine PMSO lediglich aus Variablen und einer Zielfunktion, ist diese Frage leicht zu beantworten, da die Bewertung der Position über das Optimierungsziel (also ob eine Maximierung oder Minimierung stattfindet) eindeutig bewertet werden kann. Dies ist allerdings nicht mehr der Fall, wenn neben einer Zielfunktion weitere Bedingungen beachtet werden sollen. Dann kann es zu Zwischenlösungen kommen, in denen einerseits die Zielfunktionsauswertung besser ist als zuvor, die Nebenbedingungen aber stärker missachtet werden, oder andererseits die Nebenbedingungen weniger verletzt werden, wobei die Zielfunktion schlechter wird. Für die Bewertung einer Position muss also eine genaue Anweisung vorhanden sein. Diese ist in Form eines `StateComparator` Interfaces umgesetzt. Die Wahl einer Bewertungsmethode wird über die Instanziierung eines konkreten Zustandskomparators gesteuert. Der StandardComparator wertet ausschließlich die Zielfunktion aus und berücksichtigt keine weiteren Restriktionen. Sollen diese in eine Optimierung mit eingehen, muss eine andere den `StateComparator` implementierende Klasse verwendet werden. In der Literatur ist eine Vielzahl an unterschiedlichen Bewertungsvarianten zu finden, in der vorliegenden Arbeit wurde der `EpsilonLevelComparator` integriert, welcher mit der sog. ε-Level Methode arbeitet ([Takahama und Sakai, 2006], [Fan et al., 2016]). Bei einem Vergleich im Sinne der ε-Level Methode entscheidet ein positiver Parameter ε darüber, ob sich die Partikel in Richtung besserer Zielfunktionswerte bewegen sollen oder in Richtung einer geringeren Verletzung der Nebenbedingungen. Neben der Zielfunktion (`Fitness`) wird hierfür auch ein `Violation` Objekt benötigt, welches für die Bemessung der Restriktionsverletzung über eine sog. Straffunktion zuständig ist. Für die Berechnung der Straffunktion sind zwei Varianten implementiert, welche über die Enumeration `ViolationType` vorgegeben werden können. Eine umfassende Beschreibung der *ε-Level* Methode und ihrer Varianten soll an späterer Stelle des Kapitels erfolgen. Die Auswertung der aktuellen Partikelpositionen findet bei der PMSO in jedem Iterationsschritt statt. Es bleibt die Frage, wie sich die Positionen im Laufe der Iteration ändern, um eine Suche durch den Entwurfsraum zu ermöglichen. Hierfür ist die Klasse `Search` implementiert. Sie steuert abhängig von der gewünschten Suchart die Geschwindigkeiten der Partikel und dadurch implizit deren Position. Dabei wird die Position eines Partikels in jedem Zeitschritt über die aktuelle Geschwindigkeit berechnet, die Geschwindigkeit wiederum wird über das `Search` Objekt berechnet. Die neue berechnete Geschwindigkeit setzt sich aus Anteilen der eigenen besten Position, der besten Position des Schwarms und der globalen bisher gefundenen besten Position zusammen. Die genaue Berechnungsmethode, hier als Suchart bezeichnet, kann über die Enumeration `SearchMethod` vorgegeben werden. Eine detaillierte Beschreibung der zur Verfügung stehenden Berechnungsmethoden erfolgt an späterer Stelle in diesem Kapitel. Sind nun für einen Partikel innerhalb der Iteration eine Position und eine Geschwindigkeit bekannt, muss überprüft werden, ob die gegebenen Randbedingungen eingehalten werden, was bedeutet, dass sich die Position innerhalb der spezifizierten Variablengrenzen befindet. Für die Überprüfung und für eine eventuell notwendige Nachjustierung ist die `BoundHandling` Klasse implementiert. Sie stellt sicher, dass sich die Partikel immer innerhalb der zulässigen Grenzen des Suchraums bewegen. In der Literatur sind zahlreiche Varianten für den Umgang mit Grenzen zu finden, in der vorliegenden Arbeit wurden für eine mögliche Variation drei Varianten berücksichtigt. Diese können über die Enumeration `BoundingMethod` eingestellt werden. Auch an dieser Stelle soll zugunsten eines kompakten Überblicks zunächst auf eine detaillierte Beschreibung verzichtet werden, welche zu einem späteren Zeitpunkt in diesem Kapitel wieder aufgegriffen wird.

Die Verwendung der PMSO Bibliothek soll so einfach wie möglich gestaltet sein, daher gibt es für den/die Anwender/-in möglichst wenige Schnittstellen. Die PMSO wird mit einer Reihe an Grundeinstellungen ausgeliefert und kann lediglich durch den Aufruf der `solve()` Methode

der `ParticleMultiSwarmOptimization` Klasse erfolgen. Sollen die Eigenschaften der PMSO modifiziert werden, können alle verfügbaren Parameter gesammelt an einer zentralen Stelle verändert werden. Die `PMSOOptions` Klasse dient hierbei als zentrale Schnittstelle für alle zu tätigenden Einstellungen.

Nachdem eine Übersicht über die Ontologie der PMSO gegeben wurde, soll nun die Berechnung der Partikelposition und Partikelgeschwindigkeit in jedem Zeitschritt näher betrachtet werden. Die Position P und die Geschwindigkeit V des Partikels i zum Zeitschritt t mit n Variablenwerten als Koordianten können definiert werden als

$$P_i^t = [x_{0,i}^t,\ x_{1,i}^t,\ x_{2,i}^t,\ ...,\ x_{n,i}^t] \tag{4.97}$$

$$V_i^t = [v_{0,i}^t,\ v_{1,i}^t,\ v_{2,i}^t,\ ...,\ v_{n,i}^t] \tag{4.98}$$

In jeder Iteration der Optimierung wird die Position jedes Partikels entsprechend seiner Geschwindigkeit aktualisiert. Wie bereits erwähnt wurde, setzt sich die Geschwindigkeit aus Anteilen des Partikels selber, des Schwarms und des Multischwarms zusammen. Es gilt

$$P_i^{t+1} = P_i^t + V_i^{t+1} \tag{4.99}$$

und

$$\begin{aligned} V_i^{t+1} = f \cdot V_i^t &+ w_{Individual} \cdot r_{Individual} \cdot (P_{bestIndividual}^t - P_i^t) \\ &+ w_{Social} \cdot r_{Social} \cdot (P_{bestSocial}^t - P_i^t) \\ &+ w_{Global} \cdot r_{Global} \cdot (P_{bestGlobal}^t - P_i^t) \end{aligned} \tag{4.100}$$

Zunächst einmal unterliegt die Geschwindigkeit jedes Partikels einer Trägheit f, welche als intrinsischer Parameter spezifiziert wird und einer Beschleunigung, welche durch drei Additionsterm erreicht wird. In diesen sind die Einflüsse des Partikels selbst (*Individual*), seines Schwarms (*Social*) und des Multischwarms (*Global*) zu erkennen. Die Einflüsse setzen sich immer aus dem Positionsdelta der jeweiligen bisher besten gefundenen Lösung $P_{bestIndividual}^t$, $P_{bestSocial}^t$ und $P_{bestGlobal}^t$ und der aktuellen Lösung des Partikels P_i^t zusammen und werden nochmals gewichtet. Die Gewichtung findet über die Gewichtungsfaktoren $w_{Individual}$, w_{Social} und w_{Global} statt, welche ebenso als intrinsische Parameter in die Optimierung eingehen. Zuletzt fehlt noch der stochastische Anteil. Dafür soll in jeder Iteration die Beschleunigung durch Zufallsterme gewichtet werden. Die individuelle, soziale und globale Beschleunigung werden also stochastisch durch einen prozentualen Anteil $r_{Individual}$, r_{Social} und r_{Global} (r für *random*) angepasst. Diese Anteilsparameter sind für jeden Partikel und jede Iteration einzigartig und bewegen sich zwischen den Werten 0 und 1.

Die Charakteristik der individuellen PMSO für ein bestimmtes Problem können von dem/der Anwender/-in über die intrinsischen Parameter gesteuert werden. Diese Parameter steuern den Grad der Exploration und Exploitation. Exploitation beschreibt in diesem Kontext die Fähigkeit der Partikel, vorhandenes Wissen auszuschöpfen und so die besten bisher gefundenen Lösungen anzusteuern. Exploration hingegen beschreibt die Fähigkeit der Partikel, den gesamten Forschungsraum zu erkunden und zu erschließen. Die Herausforderung für den/die Anwender/-in besteht nun darin, sinnvolle Parameterkombinationen zu finden, um ein ausgewogenes Verhältnis zwischen Exploration und Exploitation zu finden. Der Trägheitsparameter f beeinflusst dabei die Fähigkeit des Schwarms, seine Richtung zu ändern. Je niedriger f gewählt wird, desto höher ist die Konvergenz des Verfahrens, denn eine geringe Trägheit erleichtert die Ausnutzung der besten bisher gefundenen Lösungen. Eine hohe Trägheit dagegen

fördert die Schwankung um diese Lösungen herum und führt damit zu einer weiteren Exploration des Suchraums. Einen ähnlichen Einfluss haben auch die Gewichtungsfaktoren. Eine hohe individuelle Gewichtung führt zu individuellerem Verhalten der Partikel. Das Konvergenzverhalten nimmt dadurch ab, da jeder Partikel nur auf seine eigene beste Lösung konzentriert ist. Im Gegensatz dazu werden die Partikel eines Schwarms stärker von den anderen Mitgliedern beeinflusst, wenn der soziale Gewichtungsfaktor steigt. Schwarmübergreifend kann dieser gegenseitige Einfluss durch einen hohen globalen Gewichtungsfaktor gefördert werden. Ein steigender gegenseitiger Einfluss führt wiederum zu höherem Konvergenzverhalten, fördert allerdings auch das Auffinden lokaler statt globaler Lösungen.

Die bisher betrachtete Berechnung der Geschwindigkeit wird nach [Sho, 2017] mit *multiple particle swarm optimizers with information sharing* bezeichnet und ist in der vorliegenden Arbeit in der Enumeration `SearchMethod` durch den Enumerationswert `MPSOIS` gekennzeichnet. [Sho, 2017] nennt noch weitere Methoden, in denen die Gewichtungsfaktoren im Laufe der Iteration verändert werden, um ein besseres Konvergenzverhalten zu erreichen. In der vorliegenden Arbeit sind die Ansätze MPSOIWIS (*multiple particle swarm optimizers with inertia weights with information sharing*) und MCPSOIS (*multiple canonical particle swarm optimizers with information sharing*) implementiert und können über die jeweiligen Enumerationswerte SearchMethod.`MPSOIWIS` und SearchMethod.`MCPSOIS` ausgewählt werden. Die MPSOIWIS Methode modifiziert die MPSOIS Methode mithilfe eines variablen Trägheitsparameters $f(t)$, welcher vom Zeitschritt t abhängt. Abzielend auf ein besseres Konvergenzverhalten soll dieser von einem Startwert f_{Start} linear über die Zeit auf einen Endwert f_{End} reduziert werden, wobei T die maximale Anzahl der Iterationsschritte ist. Bei Wahl von MPSOIWIS gilt:

$$\begin{aligned} V_i^{t+1} = f(t) \cdot V_i^t &+ w_{Individual} \cdot r_{Individual} \cdot (P^t_{bestIndividual} - P_i^t) \\ &+ w_{Social} \cdot r_{Social} \cdot (P^t_{bestSocial} - P_i^t) \\ &+ w_{Global} \cdot r_{Global} \cdot (P^t_{bestGlobal} - P_i^t) \end{aligned} \tag{4.101}$$

$$mit \quad f(t) = f_{Start} + \frac{f_{End} - f_{Start}}{T} \cdot t \tag{4.102}$$

Dasselbe Ziel wird auch von der MCPSOIS Methode verfolgt, wobei wie bei der MPSOIS Methode ein konstanter Trägheitsparameter Φ verwendet wird, dieser aber neben der aktuellen Geschwindigkeit auch die Wirkung der äußeren Einflüsse gewichtet:

$$\begin{aligned} V_i^{t+1} = \Phi \cdot \Big[V_i^t &+ w_{Individual} \cdot r_{Individual} \cdot (P^t_{bestIndividual} - P_i^t) \\ &+ w_{Social} \cdot r_{Social} \cdot (P^t_{bestSocial} - P_i^t) \\ &+ w_{Global} \cdot r_{Global} \cdot (P^t_{bestGlobal} - P_i^t) \Big] \end{aligned} \tag{4.103}$$

Nachdem die Methoden für die Abtastung des Suchraums beschrieben wurden, soll nun darauf eingegangen werden, wie die Bewertung der aktuellen Lösung mithilfe der ε-*Level* Methode stattfindet. Wie oben beschrieben existieren hierfür zwei Varianten nach dem Vorbild von [Takahama und Sakai, 2006], deren Wahl sich in erster Linie auf die Berechnung der Straffunktion ς auswirkt. Bei Auswahl von `EpsilonMax` wird die Straffunktion durch die maximale im Zeitschritt t und Partikel i gefundene Verletzung aller j Nebenbedingungen c_j bestimmt.

$$\sigma(P_i^t) = max| \, \Delta(\, c_j(P_i^t)^{lhs},\ c_j^{rhs} \,) \, | \tag{4.104}$$

Im Gegensatz dazu wird bei Auswahl von `EpsilonSum` die Straffunktion durch die Summe aller im Zeitschritt t und Partikel i gefundener Verletzungen der J Nebenbedingungen c_j bestimmt.

$$\sigma(P_i^t) = \sum_{j=0}^{J} | \, \Delta(\, c_j(P_i^t)^{lhs},\ c_j^{rhs}\,) \, | \tag{4.105}$$

Die Auswertung der Position hinsichtlich Zielfunktionswert und Restriktionsverletzung, im Folgenden mit ε-Level-Vergleich bezeichnet, ist unabhängig von der `ViolationType` Variante und findet angelehnt an [Takahama und Sakai, 2006] mithilfe der `EpsilonLevelComparator` Klasse immer auf dieselbe Weise statt. Der ε-Level-Vergleich ist dabei definiert als eine lexikographische Ordnungsrelation, in der $\sigma(P_i^t)$ vorrangig vor $f(P_i^t)$ ist, wenn die Einhaltung der restriktiven Entwurfsraumgrenzen wichtiger ist als die Verbesserung des Zielfunktionswertes. Seien f_a und f_b ausgewertete Zielfunktionen und σ_a, σ_b die ausgewerteten Straffunktionen an den Positionen P_a, P_b. Dann gilt für den ε-Level-Vergleich mit den Operatoren $<_\varepsilon$ bzw. $\leq_\varepsilon$:

$$(f_a, \sigma_a) <_\varepsilon (f_b, \sigma_b) \Leftrightarrow \begin{cases} f_a < f_b, \text{ wenn } \sigma_a, \sigma_b \leq \varepsilon \\ f_a < f_b, \text{ wenn } \sigma_a = \sigma_b \\ \sigma_a < \sigma_b, \text{ sonst} \end{cases} \tag{4.106}$$

$$(f_a, \sigma_a) \leq_\varepsilon (f_b, \sigma_b) \Leftrightarrow \begin{cases} f_a \leq f_b, \text{ wenn } \sigma_a, \sigma_b \leq \varepsilon \\ f_a \leq f_b, \text{ wenn } \sigma_a = \sigma_b \\ \sigma_a \leq \sigma_b, \text{ sonst} \end{cases} \tag{4.107}$$

Theoretisch kann die vorgestellte und implementierte Methode nicht nur auf Nebenbedingungen, sondern gleichzeitig auch auf die Einhaltung der Variablenwertgrenzen angewandt werden. In der vorliegenden Arbeit wurde für den Umgang mit Schranken für Variablenwerte jedoch eine andere Methode integriert, welche in der `BoundHandling` Klasse aufgerufen wird. Angelehnt an das Vorgehen in [Helwig et al., 2013] wird ein Partikel mit ungültiger Position hinsichtlich der Variablenschranken repositioniert und dadurch wieder in den validen Entwurfsraum platziert. In der vorliegenden Arbeit sind drei Varianten für die Repositionierung umgesetzt. Die einfachste und am meisten intuitive Variante wird mit `BOUND` bezeichnet und setzt den Partikel bei Überschreiten der Variablengrenzen auf die Grenze der Variable $x_{k,i}^t$ selbst:

$$x_{k,i,\,updated}^t = \begin{cases} X_{k,max}, & \text{wenn } x_{k,i}^t > X_{k,max} \\ X_{k,min}, & \text{wenn } x_{k,i}^t < X_{k,min} \\ x_{k,i}^t, & \text{sonst} \end{cases} \tag{4.108}$$

Die Geschwindigkeit des Partikels bleibt dabei unberührt. Der vermeintliche Nachteil dieser Methode ist allerdings, dass es aufgrund der unveränderten Partikeldynamik sehr wahrscheinlich ist, dass der Partikel im nächsten Zeitschritt wieder über die Grenze hinaus positioniert wird. Zur Umgehung dieser Problematik wurde eine weitere Variante in der Ontologie berücksichtigt, welche auch die Geschwindigkeit modifiziert. Bei der Methode `BOUNDzero` wird die

Partikelposition genauso berechnet wie in `BOUND`, die Geschwindigkeit wird allerdings zu null gesetzt:

$$v^t_{k,i,\,updated} = \begin{cases} 0, & \text{wenn } x^t_{k,i} > X_{k,max} \\ 0, & \text{wenn } x^t_{k,i} < X_{k,min} \\ v^t_{k,i}, & \text{sonst} \end{cases} \tag{4.109}$$

In der Literatur existiert eine Vielzahl verschiedener Methoden für den Umgang mit Entwurfsraumgrenzen. [Helwig et al., 2013] geben einen umfangreichen Überblick und testen die Methoden mithilfe einer Testreihe von klassischen Optimierungsproblemen. Anhand der Ergebnisse wird eine weitere Methode besonders für Probleme empfohlen, deren Charakter nicht bekannt ist, die sog. *Reflect-Z* Methode. Für die Lösung von Packingproblemen trifft genau diese Annahme zu, da die Problemart von dem konkreten Anwendungsfall abhängt und daher nicht von vornherein feststeht. Es wird also eine universell einsetzbare Methode benötigt. In Übereinstimmung mit [Helwig et al., 2013] wurde der Enumerationswert BoundingMethod.`REFLECTzero` in die PMSO Ontologie aufgenommen. In dieser Variante wird ein Partikel nicht auf den Rand des Entwurfsraumes gesetzt, viel mehr wird die aktuelle Position an der Variablengrenze gespiegelt. Die Geschwindigkeit wird gleichzeitig zu null gesetzt. Es ergibt sich also:

$$x^t_{k,i,\,updated} = \begin{cases} X_{k,max} - (x^t_{k,i} - X_{k,max}), & \text{wenn } x^t_{k,i} > X_{k,max} \\ X_{k,min} + (X_{k,min} - x^t_{k,i}), & \text{wenn } x^t_{k,i} < X_{k,min} \text{ , sonst} \\ x^t_{k,i} \end{cases} \tag{4.110}$$

und

$$v^t_{k,i,\,updated} = \begin{cases} 0, & \text{wenn } x^t_{k,i} > X_{k,max} \\ 0, & \text{wenn } x^t_{k,i} < X_{k,min} \\ v^t_{k,i}, & \text{sonst} \end{cases} \tag{4.111}$$

Im Folgenden soll der umgesetzte PMSO Algorithmus als Ablaufdiagramm visualisiert werden. Grundsätzliche werden bei Aufruf der PMSO zwei Methoden der `Multiswarm` Klasse getriggert. Die Initialisierung (`init(p : Problem)`) und die Optimierung (`solve()`). Der Ablauf der Initialisierung ist in Abb. 4.20 dargestellt. In diesem finden Iterationen über alle Schwärme und Partikel statt, um die Positionen und Geschwindigkeiten eines jeden Partikels zu initialisieren. In den meisten Fällen wird eine Variable ohne Startwert instanziiert, in diesem Fall wird jeder Variable (welche als Koordinate der Partikelposition fungiert) ein zufälliger Wert zwischen ihrer unteren und oberen Schranke zugewiesen. Die Startgeschwindigkeit ist für diesen Fall gleich null. Ist ein Startwert für eine Variable hinterlegt, wird dieser Wert als Startkoordinate übernommen. Diese Startkoordinate gilt dann für jeden Partikel. Da sich die berechnete Geschwindigkeit für die betreffende Variable in der späteren Optimierung ausschließlich an den Koordinaten aller Partikel orientiert, muss in diesem Fall eine zufällige Startgeschwindigkeit gewählt werden (sonst gäbe es keine Varianz des Variablenwerts über die Zeit t).

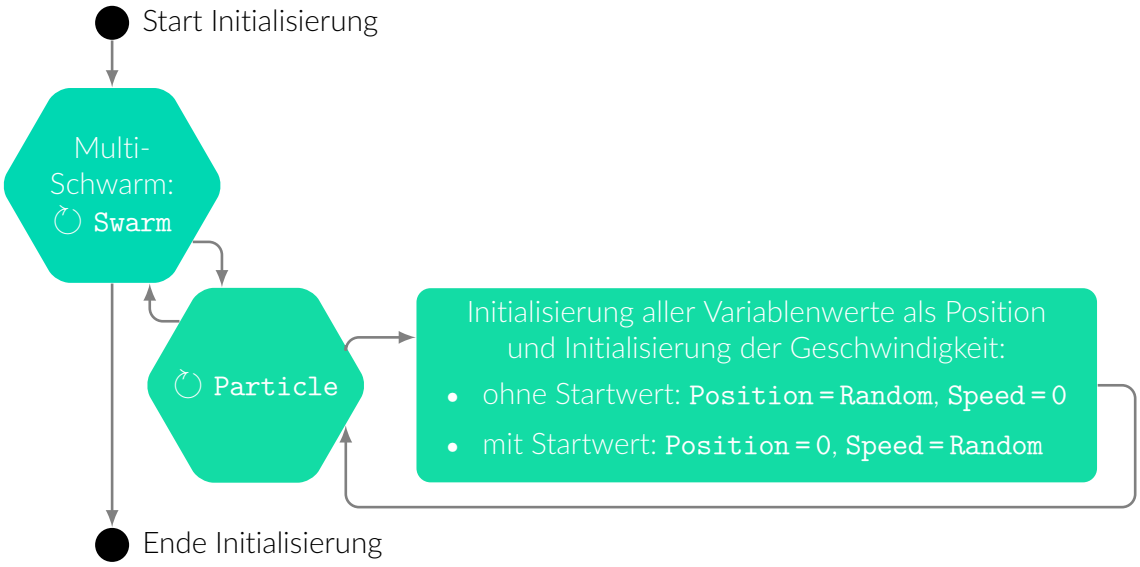


Abbildung 4.20: Algorithmus der Partikel-Multi-Schwarm-Optimierung: Initialisierung.

Nach der Initialisierung startet die eigentliche Optimierung, in der in jedem Zeitschritt jeder Partikel Änderungen unterworfen ist. Da das Optimierungsframework neben kontinuierlichen Variablen auch ganzzahlige oder boolesche Variablentypen zulässt, muss zunächst sichergestellt werden, dass diese Vorgaben eingehalten werden. Für den Umgang mit ganzzahligen und booleschen Variablen (die letztere wird als ganzzahlige Variable mit den Werten 0 und 1 behandelt) wird die in [Laskari et al., 2002] vorgeschlagene Methodik angewandt. In dieser werden ganzzahlige Variablen zunächst in kontinuierlicher Form in die Optimierung integriert und über ein stufenweises zeitschrittabhängiges Runden der Koordinaten wieder in die ganzzahlige Form überführt. In [Laskari et al., 2002] werden die Koordinaten für eine vorgegebene Anzahl an Iterationen auf 6 Nachkommastellen gerundet, eine weitere Zeitspanne auf 4 Nachkommastellen, gefolgt von einer Zeitspanne für das Runden auf 2 Nachkommastellen und schließlich eine spezifizierte Anzahl an Zeitschritten auf die ganze Zahl. Diese Vorgehensweise soll zu weniger Sprüngen während der Optimierung und dadurch zu einem besseren Konvergenzverhalten führen. Sind die Variablentypen überprüft und modifiziert, findet die Auswertung der Zielfunktion und der Verletzung der Restriktionen an der aktuellen Partikelposition statt. Diese beiden Werte bilden die Basis für die Bewertung der aktuellen Partikelposition. Wie bereits beschrieben, wird ein Komparator zur Bewertung verwendet. Dabei wird zunächst geprüft, ob die aktuelle Position gemäß dem Komparator besser ist als die bisher beste gefundene Partikelposition. Ist dies nicht der Fall, kann daraus auch geschlossen werden, dass dies weder schwarmweit noch schwarmübergreifend der Fall sein wird. Ist die aktuelle Position der bisher bekannten Partikelposition vorzuziehen, wird diese überschrieben und es kann die bisherige beste soziale Lösung des zugehörigen Schwarms verglichen werden. Führt die aktuelle Lösung zur Verbesserung der sozialen Lösung, kann wiederum die globale bisher beste entdeckte Lösung überprüft und gegebenenfalls überschrieben werden. Im nächsten Schritt findet das Update der Position und Geschwindigkeit für den nächsten Zeitschritt statt, gefolgt von einer Überprüfung, ob sich die neue Position weiterhin im validen Entwurfsraum befindet und einer eventuellen Nachjustierung der Position (und ggf. Geschwindigkeit). Diese Abfolge findet in jedem Zeitschritt für jeden Partikel jedes Schwarmes statt. Die Optimierung bricht ab, wenn die maximale Anzahl an Iterationen erreicht ist. Zusätzlich ist ein Abbruchkriterium implementiert, welches auswertet, wie oft die globale Lösung stagniert und nach einer spezifizierten Anzahl an Wiederholungen die Optimierung beendet.

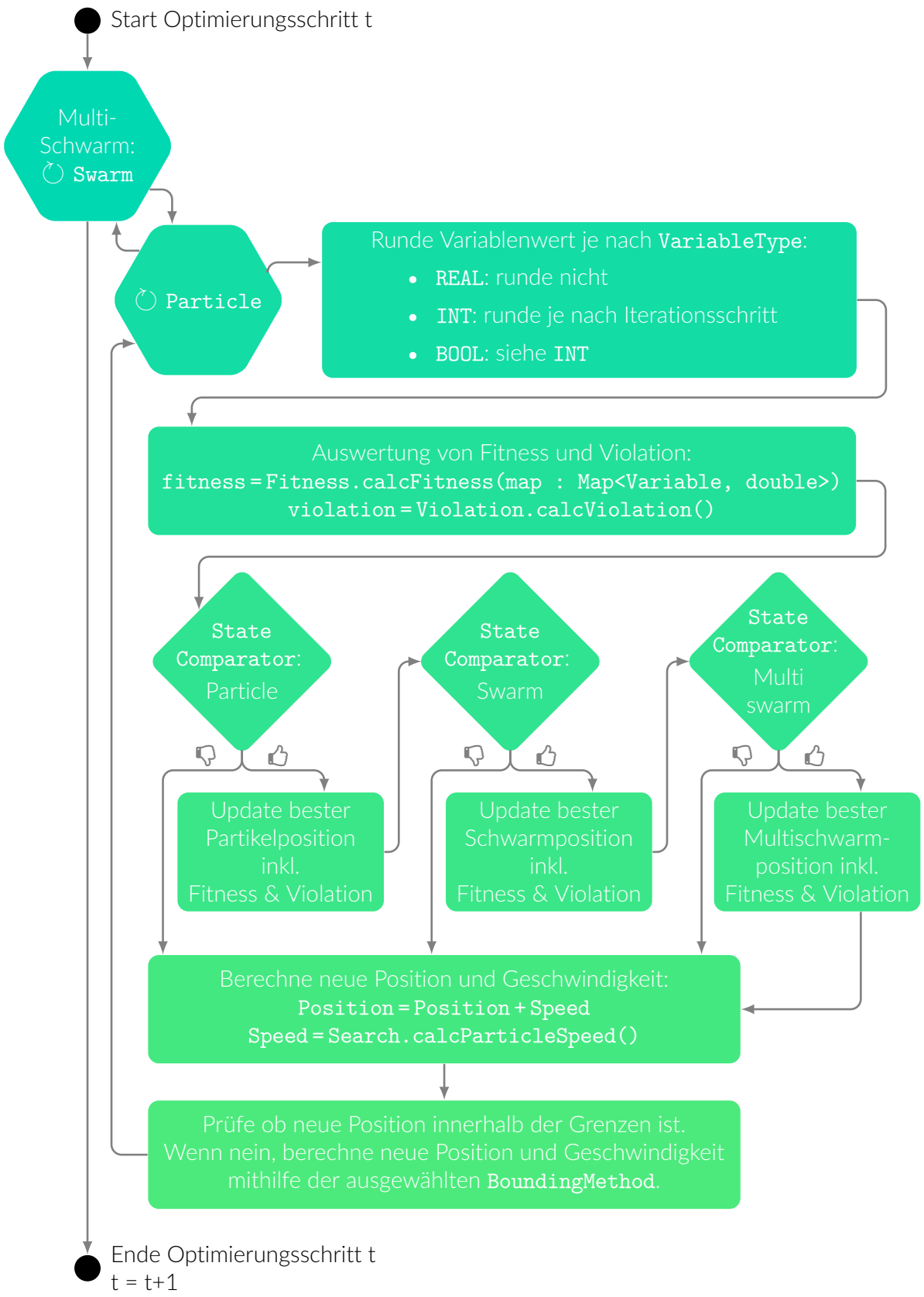


Abbildung 4.21: Algorithmus der Partikel-Multi-Schwarm-Optimierung: Iterationsschritt t innerhalb der Optimierungsschleife.

Entwurfssprache als Ziel oder Restriktion

Ein Hauptvorteil von metaheuristischen Verfahren gegenüber den deterministischen mathematischen ist deren Fähigkeit zur Verarbeitung von sog. Blackbox-Funktionen wie Programmcode oder Simulationen. Zu dieser Art von Funktion gehören auch die Entwurfssprachen selbst. Die Idee hinter dem Aufbau eines Optimierungsmoduls in Entwurfssprachen ist getrieben durch den Wunsch, dieses auch auf ganze Entwurfssprachenprojekte anwenden zu können. Daraus ergibt sich die Forderung, dass das Optimierungsmodul ermöglichen muss, dass Variablen aus Entwurfssprachen inklusive der modellierten Abhängigkeiten optimiert werden können. Gleichzeitig bedeutet dies auch, dass die physikalischen Zusammenhänge, welche bereits in Entwurfssprachen modelliert vorliegen, als Zielfunktionen oder als Nebenbedingungen in die Optimierung eingehen müssen. Vor diesem Hintergrund wurde eine Ontologie entwickelt, welche als Schnittstelle zwischen der Partikel-Multi-Schwarm-Optimierung und anderen Entwurfssprachen dient und eine Optimierung von ganzen Teilen aus Entwurssprachen unterstützt.

Zunächst soll analysiert werden, wie eine Optimierung auf eine Entwurfssprache angewendet werden kann. Dafür müssen in einem ersten Schritt die Variablen identifiziert werden, welche in die Optimierung eingehen sollen. In Entwurfssprachen kommen dafür üblicherweise die Properties aus Instanzen infrage, welche die Eigenschaften vom Produktkomponenten darstellen oder in anderen Worten die Entwurfsvariablen. Eine Forderung an die vorgestellte Ontologie ist demnach, dass sie ermöglichen muss, dass bestimmte Objekteigenschaften aus Entwurfssprachen als Variablen in der Optimierung erkannt werden und verarbeitet werden können. Hierfür wurde die Klasse `VariableNode` entwickelt. Sie koppelt die Optimierungsvariable `Variable` aus der Ontologie für die Problemdefinition (siehe Abb. 4.15) mit einer bestimmten Property einer Klasse einer bestehenden produktbezogenen Entwurfssprachenontologie. Die Kopplung einer `VariableNode` Instanz mit einer Entwurfsinstanz des bestehenden Projektes findet zur Laufzeit über eine Verlinkung der beiden Instanzen statt. Diese Verlinkung wird über die Instanziierung der `graphNode` Assoziation realisiert. Um dies zu ermöglichen, muss die in die Optimierung eingehende Produktklasse von der `VariableComponent` Klasse erben. Dafür kann der/die Anwender/-in die Klasse einfach in die eigene Ontologie importieren und die gewünschte Klasse über eine Generalisierung anbinden, wie in Abb. 4.23 dargestellt. Ist die Vererbung integriert, stehen bei der Modellierung alle Instanzen der jeweiligen Entwurfsklasse zur Verlinkung an eine `VariableNode` Instanz zur Verfügung. Bei der Modellierung ist noch zu beachten, dass ein `VariableNode` Objekt genau eine Variable darstellt und mit genau einer Eigenschaft eines Objektes vom Typ der vorliegenden Entwurfsklasse gekoppelt werden kann. Da eine Klasse bzw. ein Objekt theoretisch mehrere Properties enthalten kann, muss zur eindeutigen Zuweisung der Name der zu optimierenden Eigenschaft an die `VariableNode` Instanz übergeben werden. Mit diesen Informationen ausgestattet kann die `VariableNode` Instanz zu jedem Zeitpunkt den Wert der ausgewählten Property der zugehörigen Instanz im Graphen abrufen und vor allem überschreiben. Die letztere Fähigkeit ist von zentraler Bedeutung, denn sie ermöglicht, dass in den Iterationen der Optimierung neue von der Optimierung vorgegebene Werte in den Entwurfsgraphen geschrieben werden können, um die Entwurfssprache mit den neuen Werten aufzurufen und neue Ergebnisse zu erhalten.

Sind die eingehenden Optimierungsvariablen definiert, benötigt die Optimierung noch eine Zielfunktion. Dafür kann wie in Abschnitt 4.3.2 eine neue Zielfunktion als Subklasse der abstrakten `CodeFitness` Klasse entwickelt werden. Im Kontext der Entwurfssprachen sind die Berechnungen der Zielparameter meistens implizit im Entwurfssprachenprojekt enthalten. Das bedeutet, dass genau diese Berechnungen auch als Zielfunktion verwendet werden und die zugehörigen Zielparameter als Ergebnis der Zielfunktionsauswertung an einem bestimmten Punkt im Entwurfsraum. Diese Art der Zielfunktion soll im Folgenden mit Ziel-Entwurfssprache bezeichnet

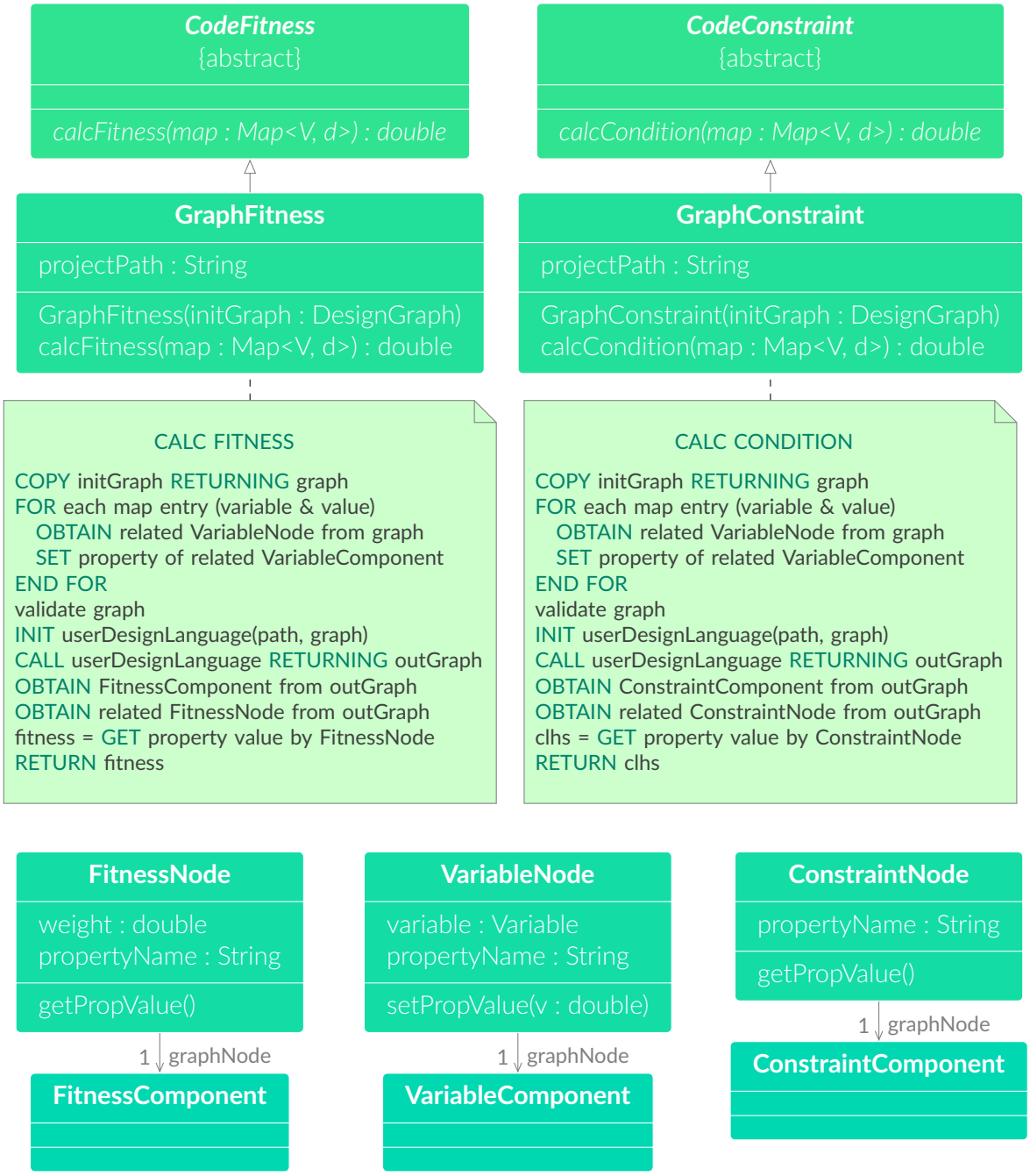


Abbildung 4.22: Ontologie der Partikel-Multi-Schwarm-Optimierung (PMSO) für die Verwendung einer Entwurfssprache als Zielfunktion oder Restriktion. Zusätzlich dargestellt sind die Hauptfunktionalitäten der Klassen GraphFitness und GraphConstraint.

werden. Wie bei den Optimierungsvariablen müssen auch die Properties welche als Zielparameter dienen sollen festgelegt werden. Das Vorgehen ist dabei das gleiche wie bei den Variablen, es werden allerdings diesmal die `FitnessNode` Klasse und `FitnessComponent` Klasse verwendet. Weiterhin kann im Sinne einer multikriterielle Optimierungen eine Gewichtung des einzelnen Zielparameters definiert werden. Für eine einfache Verwendung von ganzen Entwurfssprachprojekten als Zielfunktion ist die `GraphFitness` Klasse entwickelt und implementiert worden. Grundlage für die Funktionsfähigkeit ist, dass Teile aus Entwurfssprachen, welche als Ziel-

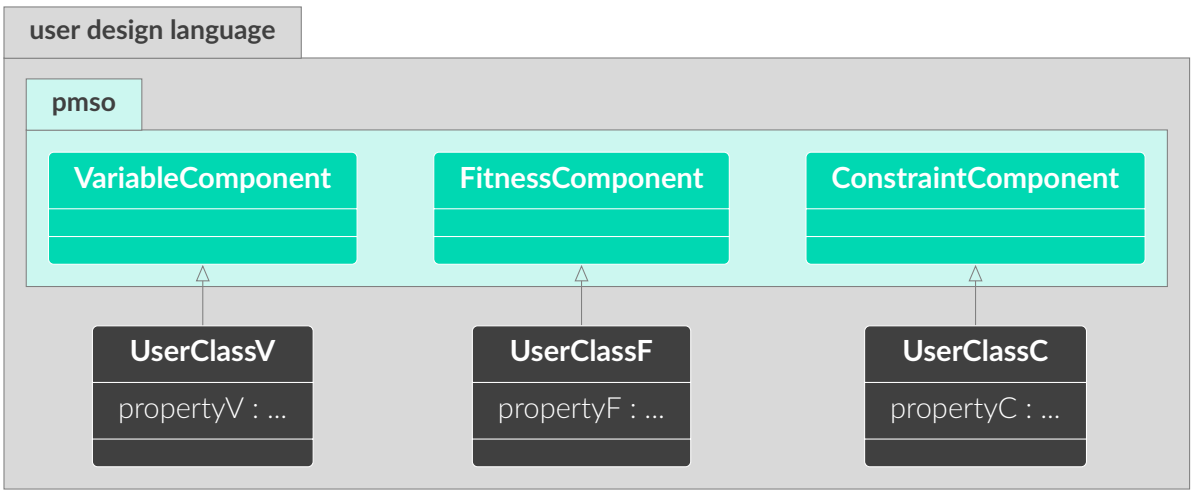


Abbildung 4.23: Anwendung der PMSO bei Verwendung einer Entwurfssprache als Zielfunktion oder Restriktion auf Klassendiagrammebene. Jede Property der User-Klassen kann als Variable in die Optimierung eingehen.

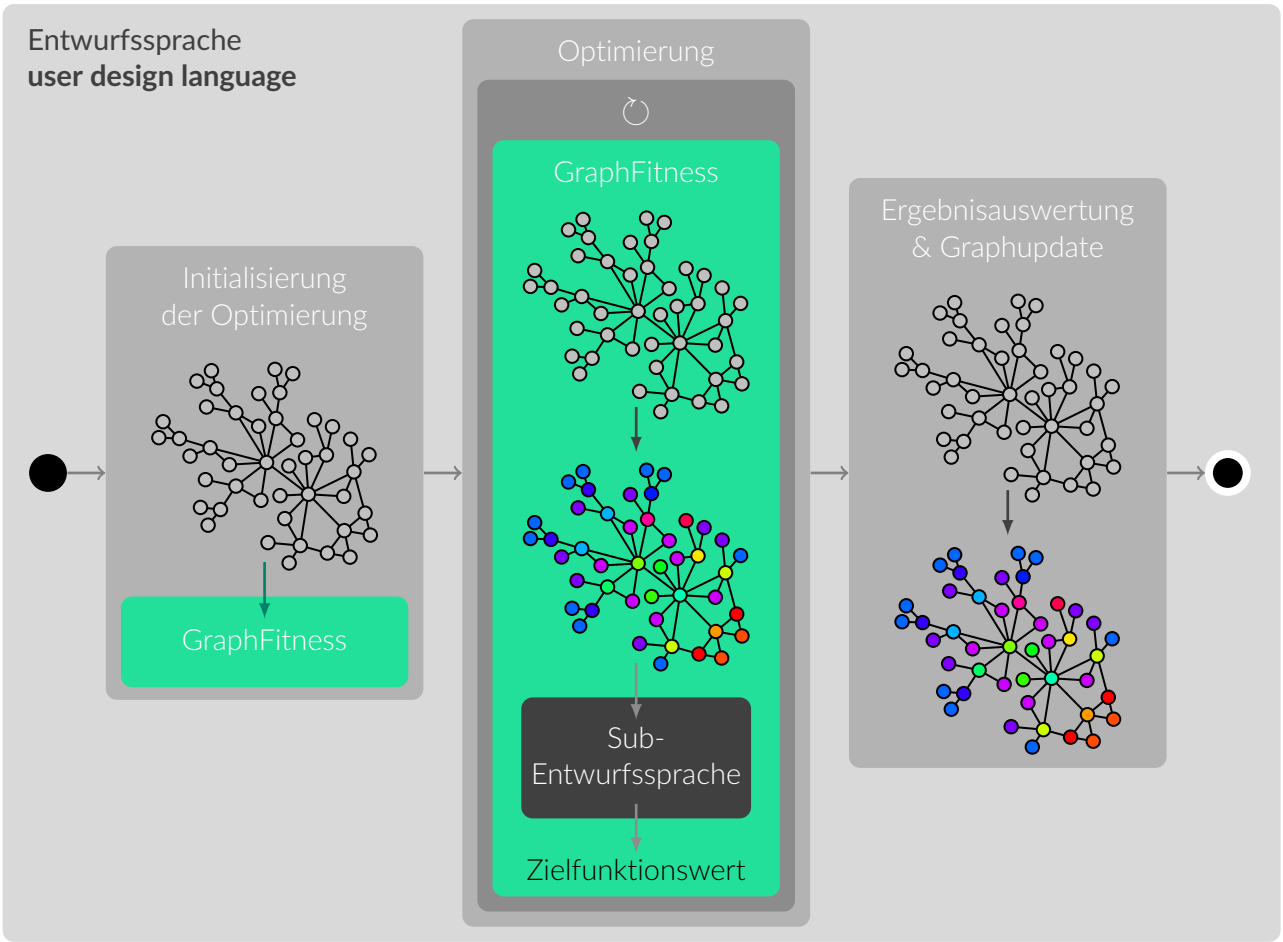


Abbildung 4.24: Anwendung der PMSO bei Verwendung einer Entwurfssprache als Zielfunktion oder Restriktion auf Programmebene. Der Ablauf der Gesamtentwurfssprache, wenn eine Subentwurfssprache als Zielfunktion genutzt wird.

Entwurfssprache agieren sollen, als eigenständige Projekte ausgelagert werden, da ein Objekt der Klasse `GraphFitness` nur auf ganze Projekte zugreifen kann. Der zugehörige Projektpfad wird bei Instanziierung der `GraphFitness` Klasse übergeben. Gleichzeitig wird ein initialer Entwurfsgraph übergeben, welcher als Basis für den späteren Input der Ziel-Entwurfssprache dient. Der Kern der Klasse besteht in der `calcFitness(...)` Methode. Diese ist in Abb. 4.22 unter der `GraphFitness` Klasse als Pseudocode dargestellt. Jeder Aufruf der Methode findet bei einer Optimierung innerhalb einer Optimierungsschleife statt. Es werden alle aktuell angenommenen Variablenwerte übergeben. In der `calcFitness(...)` Methode wird der initiale Entwurfsgraph kopiert und es werden alle darin enthaltenen Entwurfsvariablen über die `VariableNode` Klasse identifiziert sowie mit den aktuellen über die Optimierung bestimmten Variablenwerten überschrieben. Daraus ergibt sich der Inputgraph für die Ziel-Entwurfssprache. Die Ziel-Entwurfssprache wird nun mit dem aktuellen Entwurfsgraph mit iterationsspezifischen Variablenwerten ausgeführt. Nach Ausführung werden in dem zugehörigen Entwurfsgraphen alle Zielparameter über die `FitnessNode` Klasse identifiziert. Diese gehen gemäß der spezifizierten Gewichtung in eine Gesamtauswertung der Zielfunktion ein, welche schließlich ausgegeben wird. Dieselbe Vorgehensweise gilt auch für die Verwendung von Subentwurfssprachen als Nebenbedingungen, wofür die Klassen `ConstraintNode`, `ConstraintComponent` und `GraphConstraint` zur Verfügung stehen. Für ein besseres Verständnis der Anwendung ist in Abb. 4.24 der Ablauf der Gesamtentwurfssprache bei der Anwendung einer Ziel-Entwurfssprache dargestellt.

4.4 Entwurfssprache für Packingprobleme

Wie der Name bereits vermuten lässt, stellt die dritte und letzte vorgestellte Entwurfssprache, die Packagingentwurfssprache, den zentralen Kern des Packingframeworks dar. Sie baut auf den vorgestellten Entwurfssprachen für die Geometrieapproximation und die Optimierung auf und kombiniert beide zur Lösung von Packingproblemen. Die Entwurfssprache ist als Bibliothek von packingspezifischen Anwendungen implementiert und soll dem/der Benutzer/-in als Brücke zur einfachen Modellierung von Packingproblemen dienen. Die Hauptbestandteile sind eine Ontologie zur Repräsentation des Lösungsraums und der Freiheitsgrade, Klassen und Schnittstellen für die Modellierung von an den Packingprozess angepassten Geometriedarstellungen sowie Modelle für Rand- und Nebenbedingungen, welche bereits vorgefertigte Problemformulierungen für die Optimierung beinhalten. Anwender/-innen des Frameworks können zwischen verschiedenen Abstraktionsschichten wählen, um einen Einstiegspunkt zu finden, der ihren Anforderungen bestmöglich entspricht. So können sie die Packinggeometrien oder die Optimierungsmodelle selber modellieren oder auf die Packingentwurfssprache zugreifen und deren Vereinfachungen verwenden, wie in Abbildung 4.25 schematisch dargestellt ist.

Das vorliegende Kapitel ist wie folgt strukturiert. Zunächst wird ein Überblick über die allgemeine Packingontologie gegeben. Im Folgenden werden die einzelnen Hauptbestandteile detailliert erläutert. Beginnend bei der Diskretisierung des Raumes und der Freiheitsgrade in *4.4.1 Freiheitsgrade und Diskretisierung des Entwurfsraumes*, wird in *4.4.2 Geometrische Diskretisierung* die interne Darstellung der Geometrien für den Packingprozess vorgestellt, gefolgt von der Definition der packingspezifischen Nebenbedingungen in *4.4.3 Nebenbedingungen und Zwänge* wonach abschließend in *4.4.4 Ziele für den Packingprozess* auf die Repräsentation der Packingziele eingegangen wird.

Die Ontologie der Packingentwurfssprache ist Abb. 4.26 zu entnehmen. Eine Packingaufgabe ist als `Packing` Klasse abgebildet. Diese besteht aus mindestens einer zu platzierenden Komponente in Form eines `PackingComponent` Objektes, aus einem oder mehreren Zielen für das Packing in Form von `PackingObjectiveBuilder` Objekten und aus eventuellen weiteren Nebenbedingungen in Form von `PackingConstraintBuilder` Objekten. Ein Hauptbestandteil des Packings ist weiterhin das Optimierungsproblem, welches von der `Problem` Klasse aus der Ontologie der Optimierungsentwurfssprache repräsentiert wird und die Packingaufgabe abbildet. Wird eine Packingaufgabe aus Komponenten, Zielen und Nebenbedingungen aufgebaut, ergibt sich dieses Problem bei der Instanziierung der `Packing` Klasse selbstständig, und kann

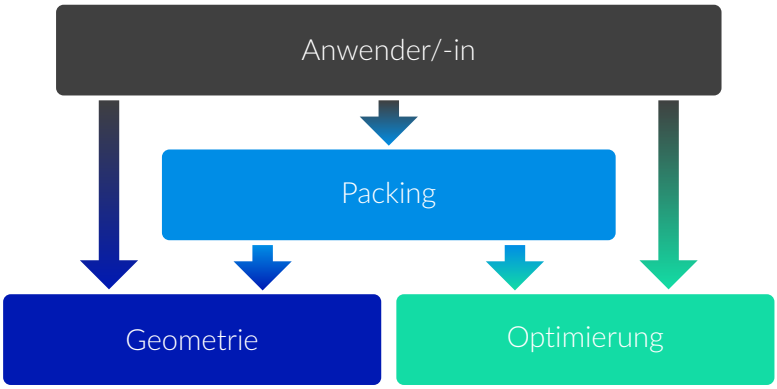


Abbildung 4.25: Struktur des Packingframeworks und Zusammenspiel der beteiligten Entwurfssprachen mit Fokus auf der Packingentwurfssprache.

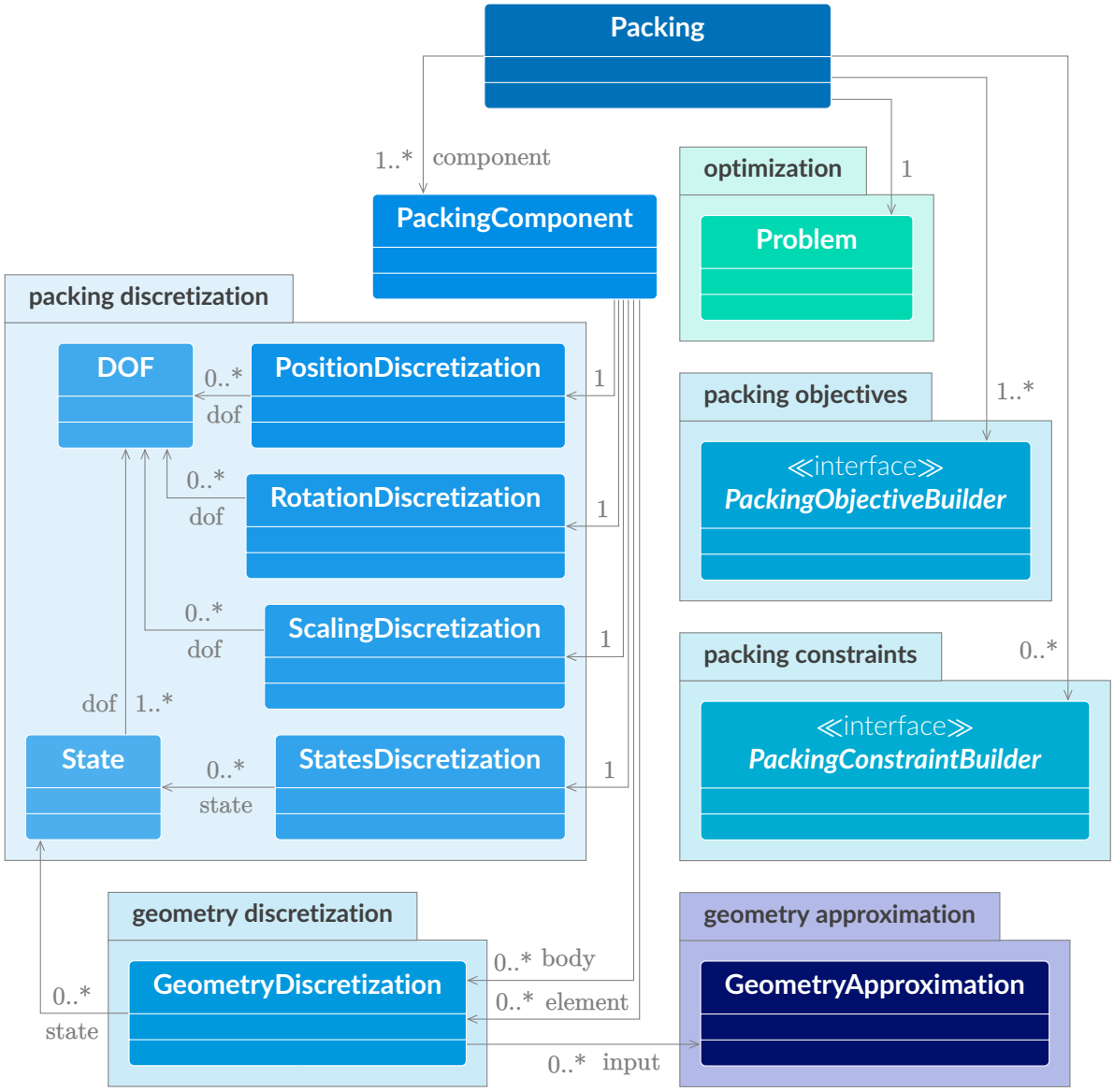


Abbildung 4.26: Ontologie der Packingentwurfssprache als UML-Klassendiagramm.

durch die Definition von weiteren Optimierungsconstraints (siehe Abschnitt 4.3.2) noch darüber hinaus angereichert werden. Die Klasse `PackingObjectiveBuilder` für die Zieldefinition wird in Abschnitt 4.4.4 näher erläutert und die Modellierung der Nebenbedingungen wird in Abschnitt 4.4.3 vorgestellt. Für eine höhere Modularität sind diese in die separaten Packages `packing constraints` und `packing objectives` ausgelagert. Für jede zu platzierende Komponente müssen deren Freiheitsgrade in Form von `DOF` Objekten (für *degree of freedom*) und die sich daraus ergebende Diskretisierung spezifiziert werden. Dabei können translatorische Freiheitsgrade über die Positionsdiskretisierung definiert werden, welche in Form einer `PositionDiscretization` Klasse implementiert ist. In derselben Weise können auch die rotatorischen Freiheitsgrade über die `RotationsDiscretization` Klasse definiert werden. Neben der Positionierung und Rotation kann weiterhin auch eine Deformation der Komponenten mithilfe der Skalierung in Form einer `ScalingDiscretization` Klasse stattfinden. Die Möglichkeit einer Deformation ist zentral für die flexible Integration des Packingprozesses in den Gesamtentwurf, da sie Rückschlüsse auf vorherige Entwurfsschritte in der Einzelteilent-

wicklung erlaubt. Ist die Entwicklung der Einzelteile abgeschlossen und keine Änderungen mehr möglich, können die Bauteilgeometrien für den Packingprozess auch eingefroren werden, indem die Skalierungsfreiheitsgrade vernachlässigt werden. Eine weitere Besonderheit der Packingentwurfssprache ist die Möglichkeit, Freiheitsgrade in Form von booleschen Zuständen abzubilden. Dafür steht die `StatesDiskretization` Klasse zur Verfügung. Die Zustände selbst werden als `State` Objekte instanziiert. Eine Erläuterung dieser Darstellung sowie die Diskussion der Vor- und Nachteile folgt in Abschnitt 4.4.1. Die Freiheitsgrade und die Diskretisierung sind in der vorgestellten Entwurfssprache im Package `packing discretization` zusammengestellt. Ein weiterer Bestandteil einer Packingkomponente ist deren räumliche Ausprägung, also deren Geometrie, welche als `GeometryDiscretization` Klasse umgesetzt ist. Dabei gibt es zwei Formen von Geometriediskretisierungen, welche berücksichtigt werden. Zum einen der materielle Körper, welcher mit anderen Komponenten zusammenstoßen kann. Dieser ist mit einer `body` Assoziation symbolisiert. Im Packingprozess muss zwischen allen geometrischen Körpern von Komponenten Kollisionsfreiheit gelten, wofür die sog. Kollisionsvermeidung eingesetzt wird. In manchen Fällen sind allerdings auch Geometrien für weiter Nebenbedingungen notwendig, welche keine materielle Struktur repräsentieren sollen. Hierfür können die `GeometryDiscretization` Objekte mithilfe der `element` Assoziation verwendet werden. Für die Definition der Geometrien steht eine Bibliothek im Package `geometry discretization` zur Verfügung. Dieses soll in Abschnitt 4.4.2 ausführlich vorgestellt werden. Es sei an dieser Stelle schon darauf hingewiesen, dass zur Geometriedefinition auch über die `input` Assoziation auf die Klassen der Ontologie für Geometrievereinfachung zugegriffen werden kann. Nicht dargestellt ist in der Abbildung außerdem, dass eine Unterscheidung der `PackingComponent` in eine zweidimensionale und dreidimensionale Variante, also die `PackingComponent2D` und `PackingComponent3D` stattfindet. Für die Repräsentation der Geometrie sind dann auch nur die zwei- bzw. dreidimensionalen Geometriediskretisierungen zulässig.

4.4.1 Freiheitsgrade und Diskretisierung des Entwurfsraumes

Die Diskretisierung des Entwurfsraumes findet wie im vorhergehenden Abschnitt erläutert über die Klassen `PositionDiscretization`, `RotationsDiscretization`, `ScalingDiscretization` oder `StatesDiscretization` statt. Jede der Diskretisierungen kann im Falle einer `PackingComponent2D` über maximal zwei bzw. im Falle einer `PackingComponent3D` über maximal drei Freiheitsgrade definiert werden. In der Entwurfssprache stehen bereits spezielle Freiheitsgradklassen zur Verfügung. Für die translatorische Positionierung sind das die Klassen `DeltaX`, `DeltaY` und `DeltaZ`. Die rotatorische Diskretisierung wird über die Freiheitsgrade `DeltaPhi` für eine Rotation um die x-Koordinatenachse, `DeltaPsi` für eine Rotation um die y-Koordinatenachse und `DeltaTheta` für eine Rotation um die z-Koordinatenachse definiert. Die Freiheitsgrade für die Skalierung werden über die Klassen `ScaleX`, `ScaleY` und `ScaleZ` definiert. Neben der Skalierung in einer Koordinatenrichtung wird auch eine uniforme Skalierung in alle Raumrichtungen über einen Freiheitsgrad der Klasse `Scale` unterstützt. Sind bestimmte Freiheitsgrade nicht gewünscht, werden sie einfach als Null-Objekte angelegt und verarbeitet. Allen Freiheitsgraden ist gemein, dass bei deren Instanziierung eine obere und eine untere Grenze definiert werden muss. Wie Abb. 4.27 zu entnehmen ist, gibt es drei mögliche Arten, um einen Freiheitsgrad zu repräsentieren. Die einfachste Variante ist die Verwendung eines kontinuierlichen Freiheitsgrades in Form eines `ContinuousDOF` Objektes. Ein kontinuierlicher Freiheitsgrad hält immer eine kontinuierliche Variable aus der Optimierungsontologie, welche in die Packingoptimierung eingeht. Eine andere Option wäre die Diskretisierung mithilfe von diskreten Freiheitsgraden, abgebildet durch die Klasse `DisreteDOF`. Ein diskreter Freiheitsgrad hält eine ganzzahlige Optimierungsvariable. Daneben wird für dessen Instanziierung eine

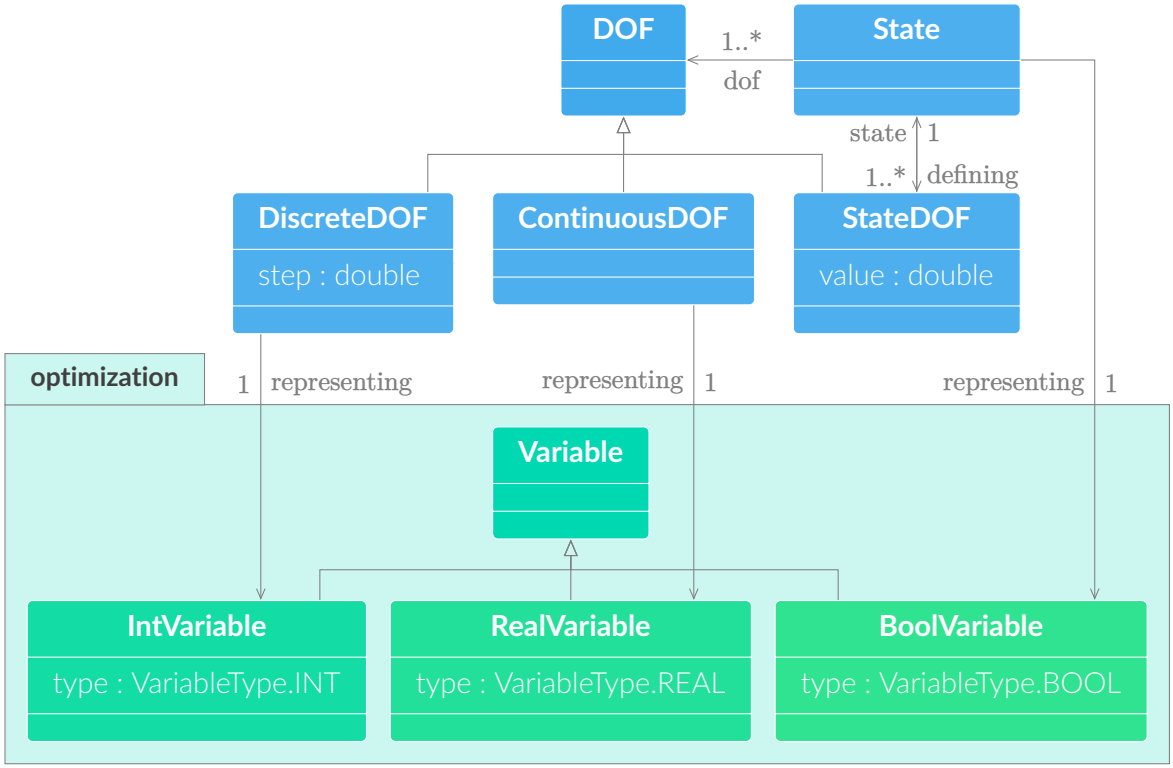


Abbildung 4.27: Ontologie der Freiheitsgrade als UML-Klassendiagramm.

Schrittweite benötigt. Die Schrittweite mal dem Wert der ganzzahligen Optimierungsvariable ergibt dann die diskreten Stützpunkte des Freiheitsgrades. Wird ein diskreter Freiheitsgrad gewählt, wird in allen zur Verfügung gestellten Modellen für die Definition von Zielen oder Nebenbedingungen statt der bloßen Variable automatisiert die ganzzahlige Variable mal der Schrittweite eingesetzt. Dadurch sind alle spezifizierten Optimierungsmodelle auch für diskrete Freiheitsgrade valide. Neben der kontinuierlichen oder diskreten Repräsentation von Freiheitsgraden gibt es noch die Möglichkeit diese über binäre Zustände zu repräsentieren, wie im folgenden Abschnitt näher erläutert werden soll.

Repräsentation der Freiheitsgrade über Szenarien

Die Wahl der eingehenden Freiheitsgrade in eine Packingoptimierung ist essenziell für die Problemklasse des Gesamtproblems. Während translatorische Freiheitsgrade keine Erhöhung der mathematischen Komplexität im Gesamtproblem zu Folge haben, können rotatorische Freiheitsgrade oder Skalierungen Auswirkungen auf den mathematischen Charakter des Gesamtproblems haben. Dies liegt vor allem an der Repräsentation der Nebenbedingungen des Packings. Rotatorische Freiheitsgrade müssen beispielsweise intern mithilfe trigonometrischer Funktionen modelliert werden. Allein die Berücksichtigung eines rotatorischen Freiheitsgrades führt also zur Nichtlinearität und Nichtkonvexität des Gesamtproblems. Die Skalierung dagegen kann nicht von allen Modellen gleichermaßen abgebildet werden. Einige der integrierten Modelle sind ausschließlich für translatorisch freie Entwurfsräume gültig und müssen bei Vorhandensein einer Skalierung durch andere, komplexere Modelle ersetzt werden. Sie haben dennoch ihre Daseinsberechtigung, da sie besonders geringe Komplexitäten aufweisen (siehe hierzu die Diskussion in *3.4.1 Lineare oder Nichtlineare Formulierungen*). Um nun die einfachen Modelle auch bei einer größeren Varianz an Freiheitsgraden verfügbar zu machen, können

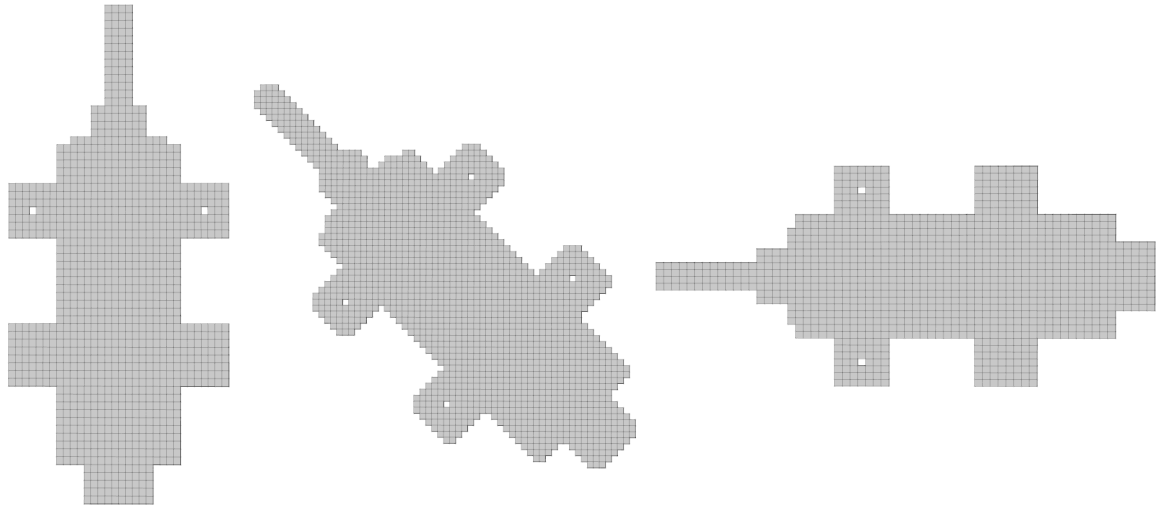


Abbildung 4.28: Zustandsbasierte Repräsentation von rotatorischen Freiheitsgraden für die Pixelisierung eines Druckwandlers. V.l.n.r.: 0°, 45° und 90°.

Freiheitsgrade auch in Form von binären Szenarien dargestellt werden. Für eine Erläuterung der Optimierung mit Szenarien sei der Leser auf Abschnitt 4.3.4 verwiesen. Werden Freiheitsgrade in Form von Zuständen dargestellt, müssen alle modellierten Nebenbedingungen und Ziele für jeden der möglichen Zustände einzeln modelliert werden. Die dadurch entstehenden *Problemsysteme* widersprechen sich dann lediglich an den Stellen, an denen die Zustände zum Tragen kommen. Um eine umfassende Gültigkeit zu erhalten, werden alle Teilsysteme automatisiert so modelliert, dass sie für die Optimierung nur dann aktiv sind, wenn die zugehörigen Zustände eintreten, und sie inaktiviert werden, wenn die jeweiligen Zustände nicht eintreten. Neben den Zielen und Nebenbedingungen müssen auch die Packinggeometrien für jeden Zustand einzeln abgebildet werden und die für jeden Zustand passende geometrische Darstellung in das Modell eingehen. Am Beispiel der Geometrien ist außerdem anschaulich zu erläutern, warum einfachere Modelle zum Einsatz kommen können. Es soll nun die Kollisionsprüfung von achsenorientierte Geometrierepräsentationen als Beispiel dienen. In Abb. 4.28 wird ersichtlich, dass für einen Rechteckverbund bzw. eine Pixelisierung lediglich die x- und y-Positionen und -Ausdehungen der einzelnen Rechtecke miteinander verglichen werden müssen, was als lineares Modell abbildbar ist. Gehen rotatorische Freiheitsgrade in die Problemstellung ein, müssten die Konturen der Rechtecke bei der Berücksichtigung als kontinuierliche Rotationsvariablen unter Einsatz von trigonometrischen Funktionen berechnet werden. Werden die rotatorischen Freiheitsgrade in Form von Zuständen repräsentiert, wird für jeden Zustand eine eigene Geometrie hinterlegt, welche immer achsenorientiert sein kann. Dadurch kann für jeden möglichen Rotationszustand eine achsenorientierte Kollisionsprüfung stattfinden.

In der Ontologie aus Abb. 4.27 ist ein solcher Zustand in Form einer `State` Klasse implementiert. Dieser wird immer über einen binären Freiheitsgrad in Form einer `StateDOF` Klasse repräsentiert. Für die Definition wird auch der Parameter `value` benötigt, welcher angibt, um welchen Zustand es sich handelt. Beispielsweise kann ein Zustand in einem zweidimensionalen Packingproblem für eine Rotation um exakt 90 Grad stehen, woraus sich der Wert 90 für den Parameter `value` ergibt. Für die Optimierung hält jeder Zustand außerdem eine boolesche Variable, die ihn repräsentiert. Insgesamt kann ein Zustand auch durch mehrere binäre Freiheitsgrade gleichzeitig definiert sein. Als Beispiel soll wieder die Rotation dienen. Im zweidimensionalen Fall orientiert sich der Zustand einer Geometrierepräsentation an dem

Zustand eines gegebenen rotatorischen Freiheitsgrades. Wird das Problem allerdings im dreidimensionalen Raum definiert, wird ein Zustand über eine Kombination der rotatorischen Freiheitsgrade spezifiziert. Für diese Kombination aus drei rotatorischen Werten kann dann die zustandszugehörige Geometrie hinterlegt werden.

Die Besonderheit bei der Darstellung von Freiheitsgraden über Zustände liegt in der Tatsache, dass sich der Charakter einer Problemdarstellung dadurch nicht verändert. Es können also mit Szenarien grundsätzlich alle Arten von Freiheitsgraden dargestellt werden, und gleichzeitig können die einfachen Modelle für Ziele und Nebenbedingungen verwendet werden. Auf der anderen Seite entsteht der Nachteil, dass die Modellierung mit Szenarien zu einer deutlichen Expansion des Gesamtproblems für die Optimierung zur Folge hat. Es muss also die Effizienzsteigerung durch die mathematische Vereinfachung der Problemdarstellung gegen den Effizienzverlust durch die ansteigende Größe des zu lösenden Systems abgewogen werden.

4.4.2 Geometrische Diskretisierung

Für die interne Repräsentation der Geometrien der Packingkomponenten steht eine in die Packingentwurfssprache integrierte Ontologie zur Verfügung. Diese darf nicht mit der zu Anfang der vorliegenden Arbeit vorgestellten Geometrieapproximation verwechselt werden, denn sie folgt grundsätzlich anderen Zielen. Die Geometriediskretisierung speichert die Geometrien in einem für die Packingverarbeitung geeigneten Format und stellt diese für die Modellierung bereit. Die Geometrieapproximation dagegen dient dem Einlesen der Geometrie und der Vereinfachung für die spätere interne Repräsentation. Abbildung 4.29 stellt die Ontologie für die Geometriediskretisierung dar. Die wichtigsten Informationen für den Packingprozess sind die Charakteristiken der Geometrien hinsichtlich ihrer Außenkontur, daran angelehnt erfolgt die Aufteilung in rechtwinklige achsenorientierte (`AxisAligned`), konvexe (`Convex`), nicht-konvexe (`NonConvex`) und orbikulare (`GDOrbicular`) Formen. Darüber hinaus orientiert sich die weitere Klassifikation an den geometrischen Merkmalen, welche entscheidend für die Wahl der Methoden zur Formulierung der Nebenbedingungen für das Packing sind. Zum einen betrifft das die Dimensionalität der Geometrie, welche in planare Geometrien (`Planar`) und dreidimensionale Geometrien (`Spatial`) unterteilt wird. Zum anderen betrifft das die Multiplizität der Form, wobei die Interfaces `GDSingle` und `GDCompound` für die Kategorisierung in einteilige und mehrteilige Geometrien zur Verfügung stehen. Am Beispiel der Kollisionsprüfung soll die erforderliche Klassifikation anhand der Merkmale kurz erläutert werden. Nimmt man das mathematische Modell einer Kollisionsprüfung für zweidimensionale Polytope, ist zu beachten, dass es unzureichend für den dreidimensionalen Fall ist. Für den dreidimensionalen Fall ist eine Erweiterung notwendig, was zwangsläufig zu einem neuen Modell führt. Je nach Zugehörigkeit eines Objektes zur Klasse `Planar` oder `Spatial` muss also eine andere Methode aufgerufen werden. Dasselbe Verhalten gilt auch für die grundlegende geometrische Einteilung der Außenkonturen, da für jede Konturvariante eine eigene Kollisionsprüfmethode hinterlegt werden muss. Die Multiplizität der Form dagegen wirkt sich insofern aus, als statt lediglich einer, parallel mehrere einzelne Kollisionsüberprüfungen stattfinden müssen.

Die Modellierung der Geometrie findet über die Instanziierung von Geometrieobjekte aus den genannten Klassen statt. Die Konstruktoren sehen eine Übergabe von `GeometryApproximation` Objekten vor, aus denen dann die Formen abgeleitet werden. Die einzelnen konkreten Klassen überschreiben dabei die jeweiligen Konstruktoren der `GDSingle` und `GDCompound` Interfaces, um eine korrekte Paarung von Geometrieapproximationen zu Geometriediskretisierungen zu gewährleisten, was aus Gründen der Übersichtlichkeit in der Abbildung vernachlässigt wurde. Für die Erstellung einer einzelnen Geometrie als `Single`-Objekt kann ein Objekt der Klasse

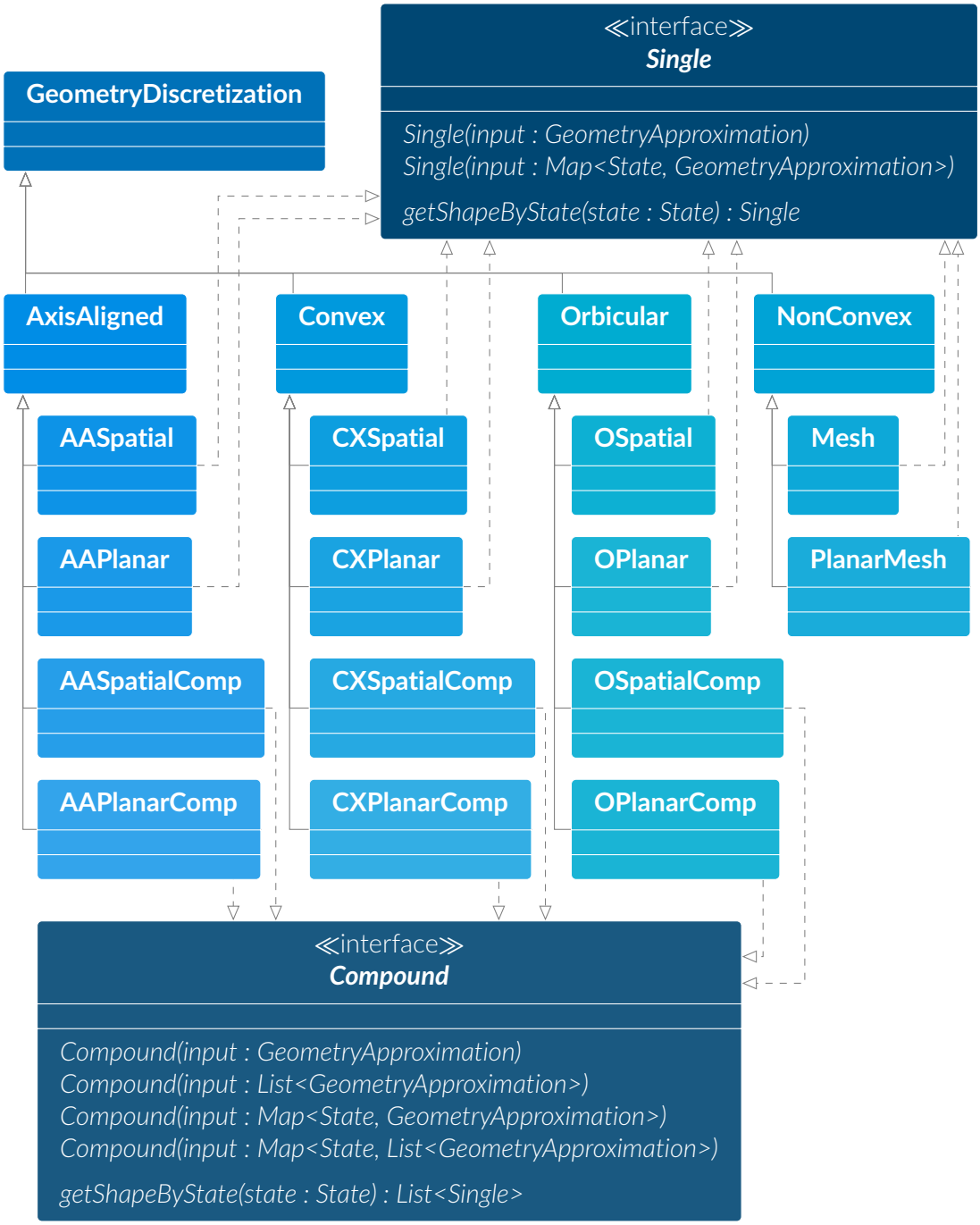


Abbildung 4.29: Ontologie als UML-Klassendiagramm für die Diskretisierung der Geometrie in der Packingentwurfssprache.

`GeometryApproximation` an den Konstruktor übergeben werden, für die Erstellung einer zusammengesetzten Geometrie als `Compound`-Objekt kann eine Liste an `GeometryApproximation` Objekten übergeben werden oder ein einzelnes, wenn dieses implizit eine ganze Gruppe von Geometrien repräsentiert. Dieser Umstand ist in Abb. 4.30 genauer dargestellt. Dort wird auch ersichtlich, wie durch Überschreibung der Konstruktoren sichergestellt wird, dass nur geeignete `GeometryApproximation` Objekte übergeben werden können. Eine Besonderheit des vorliegenden Packingframeworks ist außerdem die im vorigen Abschnitt erläuterte zustandsabhängige Diskretisierung von Freiheitsgraden. Diesen Fall berücksichtigt die Geometriediskreti-

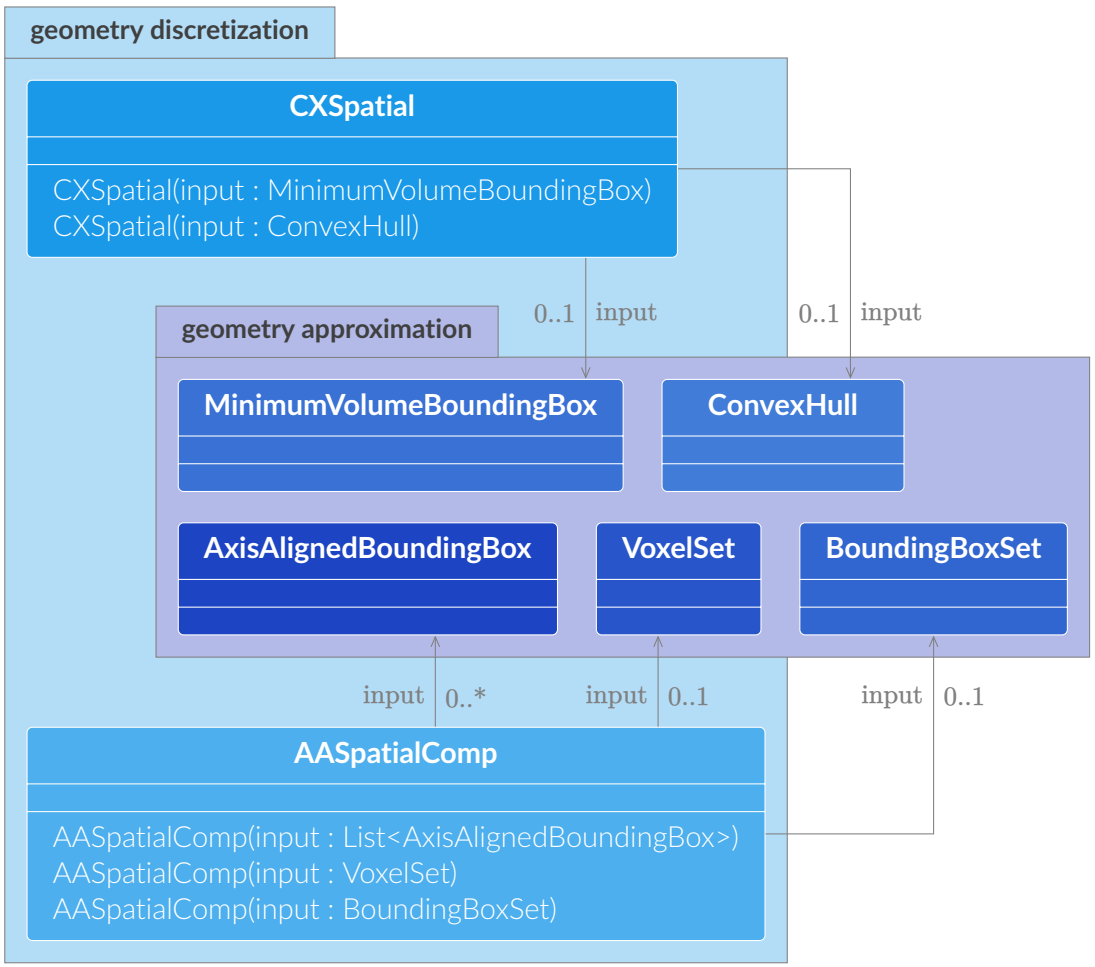


Abbildung 4.30: Geometriediskretisierung mithilfe der Geometrieapproximation am Beispiel von dreidimensionalen achsenorientierten und konvexen Geometrien.

sierung über eine Map-Datenstruktur, in der die paarweise Zuordnung von Zuständen zu den zugehörigen Geometrierepräsentationen stattfindet. Der Zugriff auf die einzelnen Geometrierepräsentationen findet über die `getShapeByState()` Methode statt, wobei bei deren Aufruf spezifiziert werden muss, zu welchem Zustand eine Geometrie abgerufen werden soll. Werden keine Zustände verwendet, kann ein *null*-Objekt übergeben werden. Trotz der grundlegenden Idee, zuerst eine Geometrie einzulesen, um sie mithilfe der Geometrieapproximation zu vereinfachen und dann in ein vom Packing lesbares Format umzuwandeln, muss diese Reihenfolge nicht zwingend eingehalten werden. Natürlich kann der/die Anwender/-in die Geometrieobjekte auch jederzeit durch Instanziierung einer leeren Klasse und das nachträgliche Befüllen der Klassenparameter von Hand erstellen.

In der entwickelten Packingentwurfssprache existieren packingsspezifische Methoden für alle geometrischen Formen, sollen diese allerdings als mathematische Formulierungen für eine deterministische mathematische Optimierung vorliegen, können nur rechtwinklige, achsenorientierte, konvexe und orbikulare Geometrien verarbeitet werden. Nicht-konvexe Geometrien müssen zunächst in eine der drei genannten Formen überführt werden. Hierfür können die Vereinfachungen aus der Ontologie der Entwurfssprache für Geometrieapproximationen (siehe Abschnitt 4.2.2) verwendet werden.

4.4.3 Nebenbedingungen und Zwänge

Der folgende Abschnitt ist den im Framework integrierten packingspezifischen Nebenbedingungen und Zwängen gewidmet. In Abschnitt 4.3.2 wurden bereits Nebenbedingungen für eine Optimierung eingeführt. Im Gesamtmodell dienen beide demselben Zweck, der Abbildung von geometrischen oder physikalischen Restriktionen und der damit einhergehenden Einschränkung des Entwurfsraums. Während die Constraints aus der Optimierungsontologie elementare und zum Teil rein mathematische Restriktionen darstellen, sind allgemeine Packingbedingungen in den meisten Fällen komplexer und müssen daher durch eine Kombination der elementaren Restriktionen aufgebaut werden. Um die Modellerstellung zu vereinfachen, wurde eine Reihe an Klassen entwickelt, welche packingspezifische geometrische oder physikalische Randbedingungen repräsentieren. Diese bestehen intern aus einer Zusammensetzung von elementaren mathematischen Constraints, welche mithilfe der Optmierungsentwurfssprache verarbeitet werden können. Aus dieser Struktur heraus werden die Packingbedingungen im Framework als sog. ConstraintsBuilder bezeichnet. Diese können Abb. 4.31 entnommen werden. Über allen spezifischen Bedingungen steht das Interface `PackingConstraintsBuilder`, welches die Gemeinsamkeiten aller `ConstraintsBuilder` Klassen spezifiziert, nämlich das Beinhalten von bei der Modellierung potenziell notwendigen Ersatzvariablen und die Möglichkeit des Abrufs aller Optimierungs-Constraints. Die Methoden zum Abruf der problemspezifischen Constraints werden bedingungsspezifisch in den konkreten, das Interface implementierenden, Klassen überschrieben. Ebenso werden die konkreten Klassen um die jeweiligen Listen erweitert, welche die problemspezifischen Constraints sortiert nach ihrem Typ enthalten. Grundsätzlich folgt die Struktur der Packingbedingungen ansonsten der grundlegenden Struktur der Nebenbedingungen in der Optimierungsontologie (siehe Abschnitt 4.3.2). Es erfolgt also über eine Vererbungshierarchie eine Unterteilung in vier grundlegende Klassen, die mathematisch abbildbaren Bedingungen, darunter die linearen (`LinearConstraintBuilder`), quadratische (`QuadraticConstraintBuilder`) und nichtlinearen (`NonLinearConstraintBuilder`) sowie die in Programmcode definierten Bedingungen (`CodeConstraintBuilder`). Dabei signalisiert die Bezeichnung der abstrakten Oberklasse immer den komplexesten Anteil, d.h. dass elementare Constraints dieser Art enthalten sind, aber durchaus auch Constraints mit weniger komplexem Charakter für die Modellbildung verwendet werden. Diese Strukturierung orientiert sich an der Optimierungsontologie, vor allem deshalb, um anschaulich darzustellen, welche Optimierungsmethoden später verwendet werden können. Im entwickelten Framework sind die als am wichtigsten eingestuften Nebenbedingungen implementiert, darunter die Kollisionsvermeidung, die geometrische Inklusion und die Einhaltung von gegebenen Abständen (siehe Abb. 4.31). Nicht abgebildet sind weitere implementierte Nebenbedingungen, die nicht zwingend als packingspezifisch einzuordnen sind und viel mehr als Hilfsbedingungen innerhalb der eigentlich zu repräsentierenden Zwänge Verwendung finden, wie die Berechnung des Betrags oder die Auswahl eines Minimums oder Maximums aus einer vorher spezifizierten Liste an Entwurfsparametern.

Der Charakter einer zu repräsentierenden Packingbedingung (also der mathematische Grad oder ob eine Darstellung als Code erforderlich ist) im Framework wird wesentlich durch drei Eigenschaften bestimmt, die Repräsentationsform der Geometrie, die zu optimierenden Freiheitsgrade und schließlich durch den mathematischen oder physikalischen Charakter der realen Bedingung selbst. Oft gibt es auch für gleiche Anforderungen verschiedene Möglichkeiten, eine Packingrestriktion zu repräsentieren. Zum besseren Verständnis soll dies anhand der Kollisionsvermeidung kurz näher erläutert werden, wobei an dieser Stelle darauf hingewiesen wird, dass für die meisten dargestellten implementierten Bedingungen eine separate detaillierte Beschreibung in den nachfolgenden Kapiteln der vorliegenden Arbeit folgt. Eine

Abbildung 4.31: Ontologie für die Modellierung von packingspezifischen Nebenbedingungen.

allgemeine Kollisionsvermeidung kann für beliebige konvexe Polytope in mathematischer Form mit quadratischem Grad erfolgen. Diese wird in der vorliegenden Ontologie durch die Klasse `QuadraticCollisionAvoidance` repräsentiert. Sollen nun rotatorische Freiheitsgrade als kontinuierliche Variablen berücksichtigt werden, muss die Drehung in der Formulierung dargestellt werden, wofür trigonometrische Funktionen vonnöten sind. Da trigonometrische Funktionen im allgemeinen nichtlinear sind, steht hierfür die Klasse `NonLinearCollisionAvoidance` zur Verfügung. Sollen im Gegenzug ausschließlich translatorische Freiheitsgrade als freie Parameter optimiert werden, kann die Kollisionsvermeidung mithilfe der `LinearCollisionAvoidance` Klasse sogar in linearer Form abgebildet werden. Hieraus lässt sich die Abhängigkeit des Charakters der Packingbedingung von den Freiheitsgraden erkennen. Es soll nun eine andere Geometrie verwendet werden. Anstatt der Polytope sollen orbikulare Geometrien, also Kreise oder Kugeln, zur Repräsentation der Komponentengeometrien verwendet werden. Für diesen Fall steht keine lineare Formulierung mehr zur Verfügung und es muss eine Kollisionsvermeidung mit mindestens zweitem Polynomgrad verwendet werden, und folglich die Klasse `QuadraticCollisionAvoidance`. Grundsätzlich ist die Kenntnis über die Verteilung der verschiedenen Bedingungen in der Packingontologie von Vorteil, wenn auch nicht zwingend erforderlich, zumindest wenn der/die Anwender/-in den Charakter des Problems nicht an ein vorher definiertes Optimierungsverfahren anpassen möchte. Ist für eine möglichst einfache Nutzung eine bloße Zusammenstellung von Packingproblemen ohne Kenntnis über die elementaren Eigenheiten des resultierenden Gesamtproblems gewünscht, unterstützt außerdem die im Folgenden vorgestellte Methodik und Zusatzontologie die Modellierung weitestgehend.

Das Strategy Entwurfsmuster für Packingbedingungen

Wie im vorigen Abschnitt angedeutet, existieren für die Anwendung der Packingentwurfssprache zwei Szenarien. Im ersten soll ausgehend von einem verfügbaren Optimierungsverfahren eine dazu passende Problemformulierung stattfinden. Für diesen Fall geben die Bezeichnungen der Klassen der Ontologie für packingspezifische Nebenbedingungen Auskunft über das Wesen der einzelnen Formulierungen. Der/die Anwender/-in benötigt allerdings ein ausreichendes Expertenwissen über die Eigenheiten der geometrischen Darstellungen und der Freiheitsgrade und einen umfassenden Überblick über alle zur Auswahl stehenden Klassen. Außerdem muss er/sie im Notfall in der Lage sein, nicht in der gewünschten Form vorliegenden Packingbedingungen selber aus elementaren Constraints zusammenzusetzen. In einem anderen Szenario bestehen keine Einschränkungen bezüglich des Optimierungsverfahrens (beispielsweise wenn für die Optimierung auf die interne Metaheuristik zugegriffen werden soll) und es soll eine möglichst schnelle Problemdefinition stattfinden. Diese zweite Variante wird durch eine Zusatzontologie unterstützt, welche mithilfe des *Strategy* Entwurfsmusters [Gamma et al., 1996] in der Entwurfssprache umgesetzt wurde. Im *Strategy* Entwurfsmuster wird eine Strategie über ein Interface umgesetzt und bildet eine Schnittstelle für alle das Interface implementierenden konkreten Strategien. Ein Kontext weist zudem eine Variable des Strategie-Interfaces auf, welche bei Bedarf mit einer Referenz auf das konkrete *Strategy* Objekt belegt wird. Die Verwendung findet über einen Klienten statt, welcher dynamisch zur Laufzeit entscheidet, welches konkrete Strategieobjekt verwendet werden soll. Abbildung 4.32 stellt das umgesetzte *Strategy* Entwurfsmuster für die Kollisionsvermeidung dar. Die Klasse `CollisionAvoidanceBuilder` stellt den Klienten dar, welcher durch den/die Anwender/-in aufgerufen wird. Dieser erstellt ein `CollisionAvoidanceContext` Objekt und spezifiziert zur Laufzeit die für die vorliegenden Geometrierepräsentationen und Freiheitsgrade passende konkrete Ausprägung der Kollisionsvermeidung (also die konkrete Strategie). Dieses vereinfachte Vorgehen ist dadurch sehr anwenderfreundlich, auch weil das schlussendlich spezifizierte Gesamtproblem selbst erkennt,

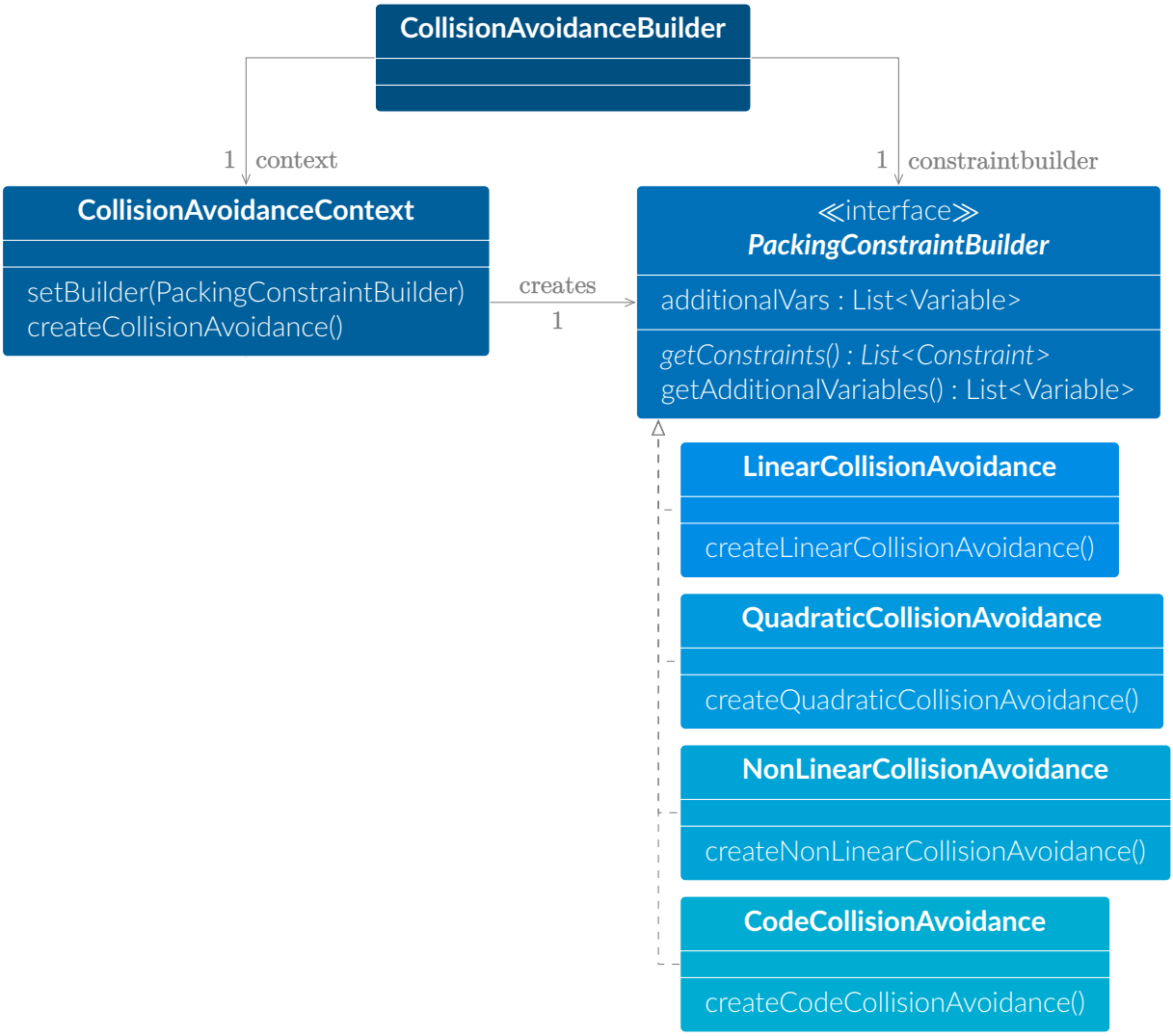


Abbildung 4.32: Strategy Entwurfsmuster für die Erstellung von Packingbedingungen am Beispiel der Kollisionsvermeidung.

von welchem Typ es ist und dem/der Anwender/-in darüber Auskunft gibt, welche Optimierungsverfahren für die Lösung in Frage kommen (Abschnitt 4.3.2). Aus diesen kann dann im Anschluss an die Problemdefinition ausgewählt werden. Durch das Vorhandensein einer internen Metaheuristik (Abschnitt 4.3.5) für die Optimierung kann außerdem sichergestellt werden, dass es prinzipiell nicht möglich ist, eine Problemrepräsentation zu erhalten, welche nicht optimiert werden könnte (wobei Optimierung in diesem Sinne den Prozess selbst meint und nicht das erfolgreiche Auffinden einer validen oder gar optimalen Lösung).

Das Abstract Factory Entwurfsmuster für die Erstellung von Optimierungs-Constraints

Wie bereits erläutert bestehen die packingspezifischen Nebenbedingungen aus einer Zusammensetzung von elementaren Optimierungsbedingungen. Der mathematische Charakter der eingehenden `Constraint` Objekte hängt dabei hauptsächlich von den Eigenschaften der realen und zu repräsentierenden Nebenbedingung ab, kann aber je nach Abhängigkeit von den eingehenden Freiheitsgraden selbst zur Laufzeit variieren. So kann ein eigentlich als nichtlinear spezifiziertes Problem bei Weglassen von rotatorischen Freiheitsgraden zur Laufzeit zu einem

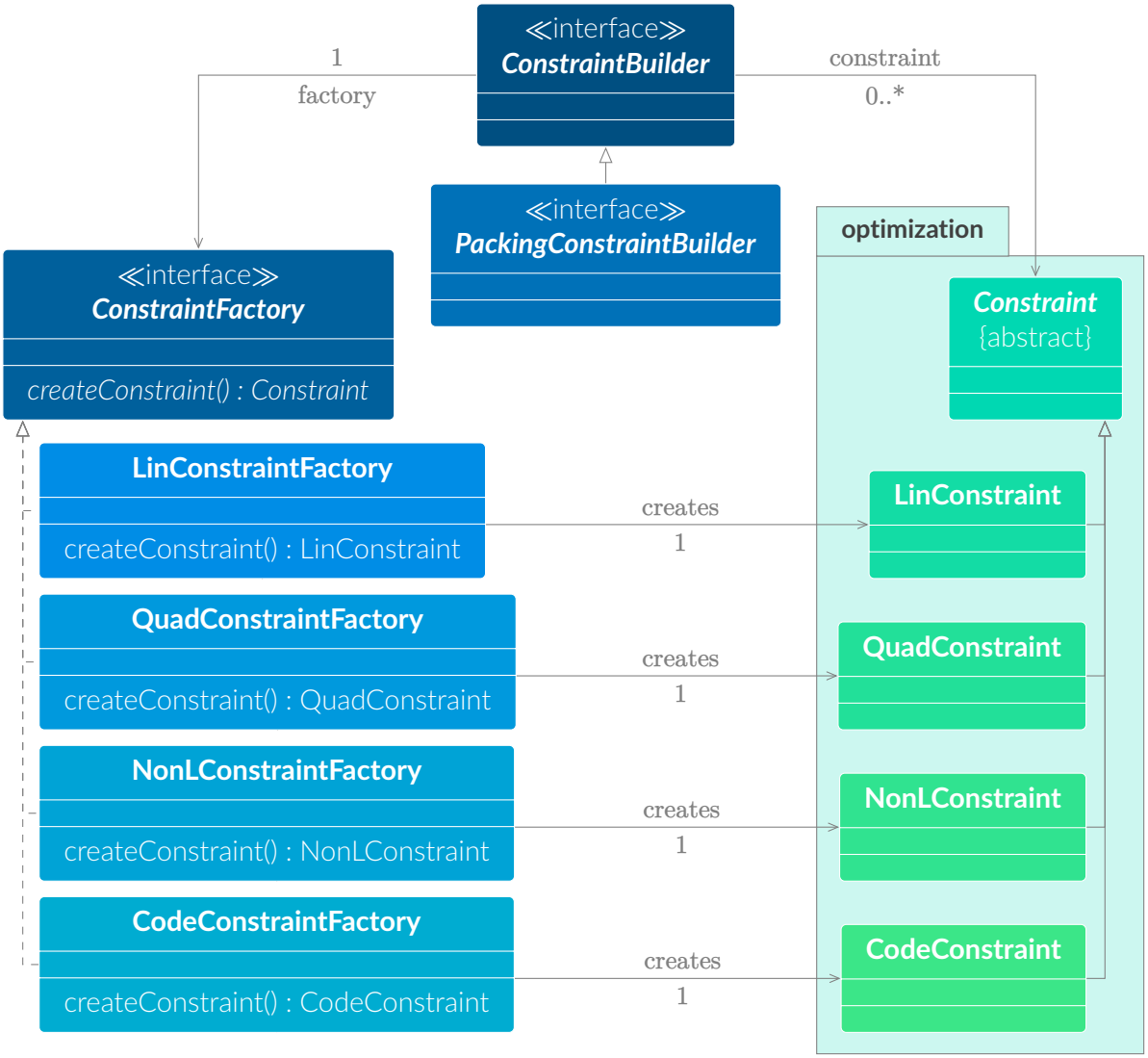


Abbildung 4.33: Abstract Factory Entwurfsmuster für die Erstellung von Optimierungs-Constraints innerhalb der PackingConstraintsBuilder Klasse.

linearen Problem werden. Innerhalb der `ConstraintBuilder` Klassen muss also eine umfassende Überprüfung aller gegebener Randbedingungen stattfinden, an der sich dann der Aufbau der elementaren `Constraint` Objekte orientiert. Im Hinblick auf eine möglichst mühelose Erweiterung der Ontologie um weitere Packingbedingungen wurde eine Hilfsstruktur entwickelt, welche die besagte Überprüfung übernimmt. In den `ConstraintBuilder` Klassen selbst kann dann einfach eine Methode aufgerufen werden, an welche alle Variablen und Terme übergeben werden, und welche abhängig von der gegebenen Situation die dazu passenden Optimierungsconstraints erstellt. Umgesetzt wird diese Struktur mithilfe des *Abstract Factory* Entwurfsmusters [Gamma et al., 1996], dessen Grundlagen bereits in Abschnitt 4.3.4 dargelegt wurden. Wie Abb. 4.33 entnommen werden kann, repräsentiert das `ConstraintBuilder` Interface den Klienten der abstrakten Fabrik, welche durch das `ConstraintFactory` Interface repräsentiert wird. Das `ConstraintBuilder` Interface hat die Fähigkeit eine gegebene Situation zur Laufzeit zu untersuchen und daraus den für die zu erstellenden `Constraint` Objekte benötigten konkreten Typen zu extrahieren. Auf dieser Basis wird ein adäquates konkretes `ConstraintFactory` Objekt erzeugt, welches wiederum die notwendigen `Constraint` Objekte

erstellt. Das `PackingConstraintsBuilder` Interface erbt diese Fähigkeit und gibt sie an alle implementierenden konkreten `ConstraintBuilder` Klassen weiter. Der Hauptvorteil dieser Struktur ist die Auslagerung der Überprüfung der notwendigen konkreten Constrainttypen in das Elterninterface. Als Folge muss dies bei einer Erweiterung der Ontologie für Packingbedingungen nicht mehr explizit von dem/der Programmierer/-in in den einzelnen erweiternden Klassen berücksichtigt werden.

Nachdem die grundlegende Struktur der Packingentwurfssprache geklärt wurde, soll nun eine Auswahl an Nebenbedingungen näher erläutert werden. Es sei an dieser Stelle bereits darauf hingewiesen, dass der Fokus hierbei auf den mathematischen Formulierungen liegt und die als Programmcode vorliegenden Darstellungen nur im Falle der Nichttrivialität einzeln detailliert besprochen werden. Dies trifft auf die Kollisionsvermeidung und die Inklusion zu, auf den Kontakt oder Abstand soll im Folgenden nicht explizit eingegangen werden. Außerdem ist zu erwähnen, dass während der Entstehung der vorliegenden Arbeit ausschließlich Nebenbedingungen integriert wurden, welche in den Anwendungsbeispielen verwendet werden sollten. Eine Erweiterung ist auf Basis des in Abschnitt 3.2 gegebenen Überblicks einfach, da für alle dort vorgestellten Nebenbedingungen bereits entwickelte Repräsentationsformen bestehen, welche in gleicher Weise auch implementiert werden können.

Kopplung von unabhängigen Freiheitsgraden

Bei der Modellierung von Packagingproblemen werden Entwurfsräume in der Regel durch die Begrenzung der unteren und oberen Variablenwerte beschränkt. Ein einfaches Beispiel hierfür wäre die Positionierung eines Bauteils in einem dreidimensionalen Raum, der in allen Raumrichtungen durch eine Ebene begrenzt ist. Dafür können einfach die drei translatorischen Variablen mit unteren und oberen Schranken belegt werden. In den meisten Fällen sind die Bauraumgrenzen allerdings nicht zu einer Koordinatenachse orthogonal, sondern liegen schräg im Raum, sodass die begrenzenden Ebenen als Linearkombination der translatorischen Variablen abgebildet werden können. In so einem Fall entstehen Zwänge durch die Kopplung der ursprünglich unabhängigen Freiheitsgrade. Ein weiterer konkreter Fall wäre beispielsweise die Positionierung von Komponenten entlang einer Ebene. Dann sind nicht mehr alle drei translatorischen Freiheitsgrade unabhängig voneinander wählbar, da zwei der räumlichen Positionierungsvariablen durch die erzwungene Positionierung auf der Ebene gekoppelt werden. Eine derartige Beschränkung wurde in der vorliegenden Arbeit bei der Verteilung von elektronischen Verteilerboxen in einem Tragflügel eingesetzt und wird in *5.2.2 Metaheuristische Positionierung von Komponenten* detailliert vorgestellt.

Kollisionsvermeidung

Eine der wichtigsten Randbedingungen für Packingvorhaben ist die Vermeidung von Kollisionen zwischen den einzelnen Objekten. In der vorliegenden Arbeit wurde eine Reihe unterschiedlicher Verfahren für die Kollsionsvermeidung entwickelt und implementiert, was eine Anpassung an ein gewünschtes Optimierungsverfahren ermöglicht. Die verschiedenen Varianten sind, abhängig von ihrem mathematischen Charakter, in den jeweiligen `CollisionAvoidance` Klassen implementiert und sollen im Folgenden vorgestellt werden.

Mathematische Kollisionsvermeidung für konvexe Polytope Die rechnergestützte Kollisionserkennung von Polytopen basiert in den meisten Fällen auf dem sog. Hyperplane Separation Theorem oder zu Deutsch auf dem sog. Trennungssatz. Dieser besagt, dass zwei disjunkte

konvexe Mengen immer so durch eine Hyperebene getrennt werden können, dass sie in unterschiedlichen Halbräumen liegen. Übertragen auf den geometrischen Fall können zwei nicht überschneidende konvexe Geometrien im zweidimensionalen Raum durch eine Gerade und im dreidimensionalen Raum durch eine Ebene voneinander separiert werden. Anwendung findet dieser Satz in der rechnergestützten Kollisionserkennung zwischen konvexen Polygonen oder Polyedern und zischen Polygonen und Kreisen bzw. zwischen Polyedern und Kugeln. Die in der vorliegenden Arbeit implementierte allgemeine Kollisionsvermeidung für konvexe Polytope basiert auf obigem Ansatz und prüft mathematisch, ob eine Hyperebene existiert, welche zwei Geometrien separiert. Zunächst soll für die nachfolgenden Erklärungen eine Reihe von Bezeichnungen für zwei Polygone P und Q eingeführt werden:

Indizes:

d: Index für die Dimension $d = 1, ..., D$ des Raumes
i: Index für den Eckpunkt $i = 1, ..., I$ des Polygons P
j: Index für den Eckpunkt $j = 1, ..., J$ des Polygons Q
k: Index für die Kante $k = 0, ..., K$ des Polygons P
l: Index für die Kante $l = 0, ..., L$ des Polygons Q
f: Index für die Fläche $f = 0, ..., F$ des Polygons P
g: Index für die Fläche $g = 0, ..., G$ des Polygons Q

Parameter:

I: Menge der Eckpunkte des Polygons P
J: Menge der Eckpunkte des Polygons Q
K: Menge der Kanten des Polygons P
L: Menge der Kanten des Polygons Q
F: Menge der Flächen des Polygons P
G: Menge der Flächen des Polygons Q
$v_{d,n}$: d-ter Eintrag des n-ten Eckpunktvektors des zugehörigen Polygons
n_d: d-ter Eintrag des Normalenvektors

Variablen:

Δ_d^{Ψ}: Verschiebung des Polygons Ψ in d-Koordinatenrichtung
s_d^{Ψ}: Skalierung des Polygons Ψ in d-Koordinatenrichtung
ρ_d^{Ψ}: Rotation des Polygons Ψ um die d-Koordinatenachse

Für zwei kollisionsfreie Polytope P und Q mit den Eckpunkten v_i aus P und v_j aus Q gilt demnach in D Dimensionen

$$\sum_{d=1}^{D}(n_d \cdot v_{d,i}) \geq u + \varepsilon \qquad \forall\ v_i \in P,\ i = 1, ..., I\ ,\quad d = 1, ..., D \tag{4.112}$$

$$\sum_{d=1}^{D}(n_d \cdot v_{d,j}) \leq u - \varepsilon \qquad \forall\ v_j \in Q,\ j = 1, ..., J\ ,\quad d = 1, ..., D \tag{4.113}$$

$$-1 \leq n_d \leq 1 \qquad d = 1, ..., D \tag{4.114}$$

wobei ε eine kleine positive Konstante darstellt. Die Konstante ε ist zur Formulierung notwendig, um zu verhindern, dass der offensichtliche Fall $0 = 0$ eintritt, welcher unabhängig

vom Kollisionszustand immer mathematisch valide ist. Um große Werte für die Konstante w zu vermeiden, die dazu führen könnten, dass die Bedingungen lediglich aufgrund von Rundungsfehlern erfüllt sind, muss n mithilfe von Gl. (4.120) begrenzt werden. Wenn nun für die obige Formulierung eine valide Lösung existiert, repräsentieren die Variablen n_d die Einträge der Normalen der separierenden Hyperebene, welche über n_d und u definiert ist. Sollten die Ungleichungen nicht erfüllbar sein, überlappen die Polytope bzw. sind in zu großer Nähe zueinander hinsichtlich der Konstante ε. Soll die Kollisionsvermeidung im Packingprozess berücksichtigt werden, müssen die obigen Nebenbedingungen das Packingproblem erweitern. An dieser Stelle ist zu erwähnen, dass die vorgestellte Kollisionsvermeidung prinzipiell auch für nichtkonvexe Polytope eingesetzt werden kann, dann allerdings implizit deren konvexe Hülle geprüft wird. Das bedeutet, dass in diesem Fall zwar auch Kollisionsfreiheit sichergestellt werden kann, der valide Entwurfsraum aber durch die Formulierung stärker eingeschränkt wird als notwendig, was mögliche globale Optima bei der Lösung ausschließen kann.

Die Klassen `QuadraticCollisionAvoidance` und `NonLinearCollisionAvoidance` stellen die obigen Formulierungen zur Verfügung, wobei die `QuadraticCollisionAvoidance` Klasse mögliche translatorische Freiheitsgrade und Skalierungsfreiheitsgrade berücksichtigt und die Klasse `NonLinearCollisionAvoidance` zum Einsatz kommt, wenn (zusätzlich) rotatorische Freiheitsgrade mit eingehen, welche eine trigonometrische Darstellung voraussetzen. Es ergibt sich demnach folgendes System für die `QuadraticCollisionAvoidance` Klasse mit den translatorischen Variablen Δ_d und den Skalierungsvariablen s_d:

$$\sum_{d=1}^{D}(n_d \cdot (s_d^P \cdot v_{d,i} + \Delta_d^P)) \geq u + \varepsilon \qquad \forall\, v_i \in P,\ i = 1, ..., I\ ,\quad d = 1, ..., D \tag{4.115}$$

$$\sum_{d=1}^{D}(n_d \cdot (s_d^Q \cdot v_{d,j} + \Delta_d^Q)) \leq u - \varepsilon \qquad \forall\, v_j \in Q,\ j = 1, ..., J\ ,\quad d = 1, ..., D \tag{4.116}$$

$$-1 \leq n_d \leq 1 \qquad d = 1, ..., D \tag{4.117}$$

Für die `NonLinearCollisionAvoidance` Klasse ergibt sich mit den translatorischen Variablen Δ_d, den Skalierungsvariablen s_d und $R_d(\rho_1, ..., \rho_D)$ als der d-ten Zeile der Rotationsmatrix $R(\rho_1, ..., \rho_D)$ mit den rotatorischen Variablen ρ_d für eine Rotation um die jeweilige Achse d:

$$\sum_{d=1}^{D}(n_d \cdot (s_d^P \cdot R_d(\rho_1^P, ..., \rho_D^P) \times v_i + \Delta_d^P)) \geq u + \varepsilon \qquad \forall\, v_i \in P,\, i = 1, ..., I\ ,\quad d = 1, ..., D \tag{4.118}$$

$$\sum_{d=1}^{D}(n_d \cdot (s_d^Q \cdot R_d(\rho_0^Q, ..., \rho_D^Q) \times v_j + \Delta_d^Q)) \leq u - \varepsilon \qquad \forall\, v_j \in Q,\, j = 1, ..., J\ ,\quad d = 1, ..., D \tag{4.119}$$

$$-1 \leq n_d \leq 1 \qquad d = 1, ..., D \tag{4.120}$$

Insgesamt führt die Formulierung auf diese Weise zu $I + J$ zusätzlichen Ungleichungen (eine Ungleichung pro Eckpunkt jedes Polytops) und $(D + 1)$ zusätzlichen kontinuierlichen Variablen ($n_1, ..., n_D$ und u) im Gesamtproblem. Sollen ausschließlich translatorische Freiheitsgrade berücksichtigt werden, kann die im Anschluss vorgestellte lineare Formulierung der `LinearCollisionAvoidance` Klasse verwendet werden.

Lineare Kollisionsvermeidung für konvexe Polytope In der Kollisionserkennung wird meistens das auf dem Trennungssatz aufbauende sog. *Separating Axis Theorem (SAT)* verwendet, welches besagt, dass sich zwei Geometrien nicht schneiden, wenn eine Achse existiert, welche die Projektionen der beiden Geometrien separiert. Dabei kann sowohl im zweidimensionalen als auch im dreidimensionalen Fall von einer Gerade gesprochen werden, da zweidimensionale Polytope sowieso durch eine Gerade voneinander getrennt werden und im dreidimensionalen Fall die Gerade senkrecht auf der separierenden Ebene steht. Auf Basis des *SAT* lassen sich für die Kollisionsprüfung von vorneherein mögliche Achsen ableiten, welche zu einer separierenden Ebene führen. Letztere sind entweder durch die Normale einer der Oberflächen der Geometrien gegeben (in 2D und 3D) oder ergeben sich aus dem Kreuzprodukt zweier Kanten der beiden Geometrien (in 3D). Das bedeutet, dass in Algorithmen zur Kollisionserkennung nur diese Varianten überprüft werden müssen. Die in der vorliegenden Arbeit implementierte lineare Kollisionserkennung besteht aus einer mathematisch gemischt-ganzzahligen Formulierung des *SAT*. Dabei werden für zwei Geometrien alle separierenden Achsen als Ungleichung modelliert und mit einer Indikatorvariable behaftet. Dadurch wird eine Reihe an Zuständen repräsentiert, in denen jeweils eine separierende Achse die beiden Polytope voneinander trennt und von denen mindestens ein Zustand wahr sein muss. Zur gemischt-ganzzahligen Darstellung von Zuständen sei auf Abschnitt 4.4.1 verwiesen. Wird die Gültigkeit einer der Ungleichheiten erzwungen, kann die Kollisionsfreiheit der zugrunde liegenden Geometrien in mathematischer Form und ohne rotatorische Freiheitsgrade sogar in linearer Form sichergestellt werden.

Die Formulierung basiert dabei auf der Veröffentlichung von [Scheithauer und Terno, 1993], in welcher der zweidimensionale Fall vorgestellt wurde. Darin wird beschrieben, dass eine separierende Achse dann besteht, wenn sie über die f-te Oberfläche des Polytops P gebildet wird und derjenige Eckpunkt v_j des Polytops Q, dessen Projektion auf den Normalenvektor der zugehörigen Ebene den kleinsten Wert annimmt (welcher auch negativ sein kann), auf der positiven Seite der Ebene liegt. Dieser Punkt soll im Folgenden als der Punkt bezeichnet werden, welcher bezüglich der Ebene die negativste Entfernung einnimmt. Voraussetzung ist dabei, dass die Ebenennormale für Polytope immer nach außen zeigt. Angenommen also eine Ebene e durch den Punkt v kann in Koordinatenform dargestellt werden als

$$e(v_1, ..., v_D) := \sum_{d=1}^{D} (n_d \cdot v_d) + h = 0 \qquad d = 1, ..., D \tag{4.121}$$

dann lautet die zugehörige mathematische Darstellung zur Kollisionsvermeidung mit den booleschen Variablen Θ_f^Q und Θ_g^P und einer ausreichend großen Konstante M

$$-\sum_{d=1}^{D} \Big(n_{d,f} \cdot (\Delta_d^Q - \Delta_d^P) \Big) - h_f - h_f^Q \leq M \cdot (1 - \Theta_f^Q) \qquad f = 1, ..., F \ , \quad d = 1, ..., D \tag{4.122}$$

$$-\sum_{d=1}^{D} \Big(n_{d,g} \cdot (\Delta_d^P - \Delta_d^Q) \Big) - h_g - h_g^P \leq M \cdot (1 - \Theta_g^P) \qquad g = 1, ..., G \ , \quad d = 1, ..., D \tag{4.123}$$

wobei

$$h_f^Q := min\Big\{ \sum_{d=1}^{D} (n_{d,f} \cdot v_{d,j}) \Big\} \qquad j = 1, ..., J \ , \quad f = 1, ..., F \ , \quad d = 1, ..., D \tag{4.124}$$

$$h_g^P := min\Big\{ \sum_{d=1}^{D} (n_{d,g} \cdot v_{d,i}) \Big\} \qquad i = 1, ..., I \ , \quad g = 1, ..., G \ , \quad d = 1, ..., D \tag{4.125}$$

und

$$\sum_{f=1}^{F} \Theta_f^Q + \sum_{g=1}^{G} \Theta_g^P = 1 \qquad j = 1, ..., J \ , \quad f = 1, ..., F \tag{4.126}$$

Für eine detaillierte Beschreibung sei der Leser auf [Scheithauer und Terno, 1993] verwiesen. Der zweidimensionale Fall wurde in der vorliegenden Arbeit in der `LinearCollisionAvoidance` Klasse implementiert und für den dreidimensionalen Fall um die Überprüfung des Kreuzproduktes erweitert. Bei der Erweiterung in 3D gilt weiterhin, dass der Punkt v_i des Polytops P, welcher bezüglich der Ebene die negativste Entfernung einnimmt, auf der positiven Seite der Ebene liegt und der Punkt v_j des Polytops Q, welcher bezüglich der Ebene die positivste Entfernung einnimmt, auf der negativen Seite der zugehörigen separierenden Ebene liegt (oder andersherum), wenn deren Normale über das Kreuzprodukt der k-ten und l-ten Kante der beiden Polytope gebildet wurde. Für zwei Polytope P und Q führt die Gesamtformulierung schließlich zu $(F + G + 1)$ zusätzlichen Ungleichungen in 2D und $(F + G + K \cdot L + 1)$ zusätzlichen Ungleichungen in 3D (eine Ungleichung pro potenzielle separierende Ebene und die Zustandsaktivierungsgleichung) sowie zu $(F+G)$ zusätzlichen booleschen Variablen in 2D bzw. $(F+G+K \cdot L)$ zusätzlichen booleschen Variablen in 3D (eine Indikatorvariable pro potenzielle separierende Ebene) im Gesamtproblem. Wird ausgenutzt, dass die booleschen Variablen Θ_f^Q und Θ_g^P nie gleichzeitig wahr sind, kann die *kompakte* Form der Modellierung gewählt werden (siehe dazu *4.3.4 Effiziente Modellierung von Szenarien*) und die Anzahl der zusätzlichen Ungleichungen reduziert sich auf $(F + G)$ bzw. $(F + G + K \cdot L)$ da die Zustandssummengleichung vernachlässigt werden kann und die Anzahl der zusätzlichen Variablen reduziert sich auf $log_2(F + G)$ bzw. $log_2(F + G + K \cdot L)$.

An dieser Stelle sei der Leser darauf hingewiesen, dass die lineare Formulierung nur zum Einsatz kommen kann, wenn keine rotatorischen Freiheitsgrade berücksichtigt werden müssen, da diese den Einsatz von trigonometrischen nichtlinearen Funktionen voraussetzen. Um die lineare Formulierung trotz vorhandener rotatorischer Freiheitsgrade einsetzen zu können, kann die Rotation in Form vom Zuständen diskretisiert werden (siehe dazu Abschnitt 4.4.1). Der Einsatz von Skalierungsvariablen verändert den mathematischen Charakter im Allgemeinen ebenso, sodass auch diese bei Anwendung der linearen Formulierungen der `LinearCollisionAvoidance` Klasse über Zustände repräsentiert werden müssen.

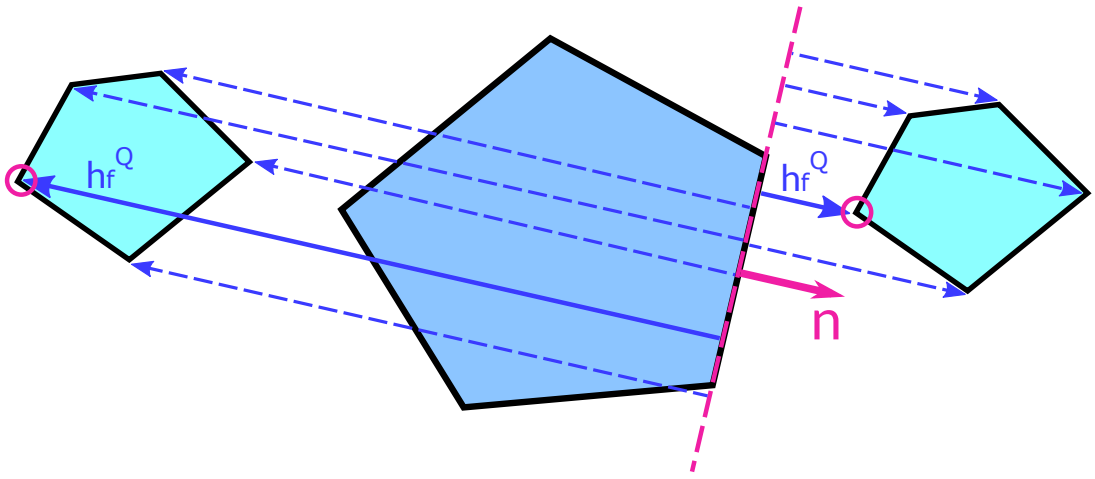


Abbildung 4.34: Lineare Formulierung der Kollisionsvermeidung konvexer Polytope in 2D.

Lineare Kollisionsvermeidung für rechtwinklige Polytope Für Rechtecke und Quader gelten im Allgemeinen dieselben Kollisionsgleichungen wie für konvexe Polytope. Es können allerdings auch Formulierungen aufgestellt werden, welche auf den rechtwinkligen Fall zugeschnitten sind und dadurch zu einer geringeren Anzahl an zusätzlichen Gleichungen und Variablen führen. Die hierfür vorgestellte Kollisionsvermeidung ist dem Leser bereits in einigen Kapiteln begegnet, da sie ein anschauliches Beispiel für die grundlegenden Vorgehensweisen bei der ganzzahlig-linearen Optimierung bietet. Aus diesem Grund soll sie an dieser Stelle nur kurz zusammengefasst werden. Für die Kollisionsvermeidung von rechtwinkligen Polytopen gilt in D Dimensionen mit den booleschen Variablen Θ_d^{PQ} und Θ_d^{QP} und einer ausreichend großen Konstante M

$$s_d^P \cdot (v_{d,i=0} + \lambda_d^P) + \Delta_d^P \leq s_d^Q \cdot v_{d,j=0} + \Delta_d^Q + M \cdot (1 - \Theta_d^{PQ}) \qquad d = 1, ..., D \tag{4.127}$$

$$s_d^Q \cdot (v_{d,j=0} + \lambda_d^Q) + \Delta_d^Q \leq s_d^P \cdot v_{d,i=0} + \Delta_d^P + M \cdot (1 - \Theta_d^{QP}) \qquad d = 1, ..., D \tag{4.128}$$

$$\sum_{d=1}^{D} \Theta_d^{PQ} + \sum_{d=1}^{D} \Theta_d^{QP} = 1 \qquad d = 1, ..., D \tag{4.129}$$

wobei $v_{d,0}$ den linken unteren Eckpunkt des zugehörigen Polytops repräsentiert und λ_d die Länge des Rechtecks oder Quaders in die Raumrichtung d. Die Formulierung führt zu $2D$ zusätzlichen Ungleichungen sowie zu $2D$ zusätzlichen booleschen Variablen im Gesamtproblem. Im Vergleich zur linearen Darstellung der Kollisionsvermeidung von allgemeinen Polytopen, reduziert sich die Anzahl der zusätzlichen Gleichungs- und Variablenanzahl dadurch, dass die Parallelität der Flächen der rechtwinkligen Polytope ausgenutzt wird und eine Ungleichung gleichzeitig für eine Fläche aus beiden Polytopen gilt. Implementiert ist die obige Darstellung in der `LinearCollisionAvoidance` Klasse und gilt sowohl für kontinuierliche translatorische Freiheitsgrade wie auch für kontinuierliche Skalierungsvariablen. Lediglich bei der Verwendung von rotatorischen Freiheitsgraden, müssen diese über Zustände repräsentiert werden.

Mathematische Kollisionsvermeidung für orbikulare Geometrien Die Kollisionsvermeidung für orbikulare Geometrien kann in mathematischer Form mithilfe eines quadratischen Systems an Ungleichungen repräsentiert werden. Dabei kollidieren zwei Kreise oder Kugeln nicht, wenn der Abstand ihrer Mittelpunkte c^P und c^Q größer ist als die Summe der Radien r^P und r^Q. Der Fall für die Berücksichtigung von translatorischen Freiheitsgraden und einer Skalierungsvariable pro Komponente ist in der `QuadraticCollisionAvoidance` Klasse implementiert und wird respräsentiert als

$$\sum_{d=1}^{D} ((s^P \cdot c_d^P + \Delta_d^P) - (s^Q \cdot c_d^Q + \Delta_d^Q))^2 \geq (s^P \cdot r^P + s^Q \cdot r^Q)^2 \qquad d = 1, ..., D \tag{4.130}$$

mit den translatorischen Variablen Δ_d und den Skalierungsvariablen s. Die Skalierung darf hierbei nur gleichförmig in alle Raumrichtungen stattfinden, da die koordinatenweise Deformation den Charakter des Problems verändern und die Formulierung damit ungültig werden würde. Sollen rotatorische Freiheitsgrade berücksichtigt werden, können dieser entweder in Form von Zuständen repräsentiert werden oder es wird die nichtlineare Variante verwendet, welche in der `NonLinearCollisionAvoidance` Klasse zu finden ist. Diese lautet

$$\sum_{d=1}^{D} ((s^P \cdot R_d^P \times c^P + \Delta_d^P) - (s^Q \cdot R_d^Q \times c^Q + \Delta_d^Q))^2 \geq (s^P \cdot r^P + s^Q \cdot r^Q)^2 \qquad d = 1, ..., D \tag{4.131}$$

mit $R_d = R_d(\rho_1, ..., \rho_D)$ als der d-ten Zeile der Rotationsmatrix $R(\rho_1, ..., \rho_D)$ mit den rotatorischen Variablen ρ_d für eine Rotation um die jeweilige Achse d.

Blackbox Kollisionserkennung Die in der `CodeCollisionAvoidance` Klasse umgesetzte Kollisionsvermeidung basiert auf der im *Design Cockpit 43*® [IILS, 2022] softwareinternen Kollisionsdetektion, welche bereits fester Bestandteil von Entwurfssprachen ist. Für die Kollisionsvermeidung wird eine Ungleichung aufgebaut, deren Konditionalteil auf die Auswertung der Kollisionsdetektion zugreift. Weiterhin wird durch die Ungleichung gefordert, dass das Auswertungsergebnis kleiner gleich 0 ist. Die Kollisionsfreiheit wird dann sichergestellt, indem eine Kollision einen positiven Wert bei der Auswertung ausgibt und keine Kollision den Wert 0. Diese auf Programmcode basierende Kollisionsvermeidung ist bei der Verwendung von deterministischen mathematischen Optimierungsverfahren nicht anwendbar. Es muss auf Optimierungsverfahren zurückgegriffen werden, welche in der Lage sind Blackbox Funktionen zu verarbeiten.

Geometrische Inklusion

Ein typischer Anwendungsfall für den Packingprozess im Ingenieurwesen ist die Platzierung von Komponenten innerhalb eines Bauraumes. Die einfachste Möglichkeit einen Bauraum vorzugeben ist über die Spezifikation einer Inklusionsbedingung, welche sicherstellt, dass alle Bauteile von einem gegebenen Bauraum umschlossen werden. Im Folgenden sollen verschiedene Möglichkeiten für die Repräsentation einer solchen Inklusionsbedingung vorgestellt werden, wobei alle Varianten auch in das entwickelte Framework integriert wurden und anwendbar sind.

Mathematische Inklusionsformulierung für konvexe Polytope Für konvexe Polytope gilt im Allgemeinen, dass jeder Punkt innerhalb eines Polytops P auch als Linearkombination der Eckpunkte v_i des Polytops ausgedrückt werden kann. Wenn das konvexe Polytop P ein konvexes Polytop Q vollständig umschließt, oder in anderen Worten das Polygon Q vollständig innerhalb des Polygons P liegt, gilt folglich, dass auch alle Eckpunkte v_j des Polygons Q durch eine Linearkombination der Eckpunkte v_i des Polygons P ausgedrückt werden können. Es gilt demnach für die Inklusion von konvexen Polygonen mit den Koeffizienten w_{ij}

$$\sum_{i=1}^{I} w_{ij} \cdot (s_d^P \cdot v_{d,i} + \Delta_d^P) = s_d^Q \cdot v_{d,j} + \Delta_d^Q \qquad \begin{array}{l} \forall\ v_i \in P,\ \forall\ v_j \in Q \\ i = 1, ..., I\ ,\ j = 1, ..., J \\ d = 1, ..., D \end{array} \tag{4.132}$$

$$\sum_{i=1}^{I} w_{ij} = 1 \qquad i = 1, ..., I\ ,\ j = 1, ..., J \tag{4.133}$$

$$0 \leq w_{ij} \leq 1 \qquad i = 1, ..., I\ ,\ j = 1, ..., J \tag{4.134}$$

wobei eine Inklusion besteht, wenn eine valide Lösung für die Werte w_{ij} gefunden wird und keine Inklusion im nicht validen Entwurfsraum für die Werte w_{ij} liegt. Für die Berücksichtigung der translatorischen Freiheitsgrade und Skalierungsvariablen ist die obige Formulierung aufgrund des quadratischen Charakters in der `QuadraticContainment` Klasse implementiert.

Für deren zulässige Anwendung müssen potenzielle rotatorische Freiheitsgrade in Form von Zuständen repräsentiert werden. Sollen die rotatorischen Freiheitsgrade als Variablen eingehen, muss die folgende Formulierung aus der `NonLinearContainment` verwendet werden

$$\sum_{i=1}^{I} w_{ij} \cdot (s_d^P \cdot R_d^P \times v_i + \Delta_d^P) = s_d^Q \cdot R_d^Q \times v_j + \Delta_d^Q \qquad \begin{array}{l} \forall\ v_i \in P,\ \forall\ v_j \in Q \\ i = 1, ..., I\ ,\ j = 1, ..., J \\ d = 1, ..., D \end{array} \tag{4.135}$$

$$\sum_{i=1}^{I} w_{ij} = 1 \qquad i = 1, ..., I\ ,\ j = 1, ..., J \tag{4.136}$$

$$0 \leq w_{ij} \leq 1 \qquad i = 1, ..., I\ ,\ j = 1, ..., J \tag{4.137}$$

mit $R_d = R_d(\rho_1, ..., \rho_D)$ als der d-ten Zeile der Rotationsmatrix $R(\rho_1, ..., \rho_D)$ mit den rotatorischen Variablen ρ_d für eine Rotation um die jeweilige Achse d. Insgesamt wächst hierdurch die Anzahl der Ungleichungen im Gesamtproblem um $(D \cdot J)$ und die Anzahl der kontinuierlichen Variablen um $(I \cdot J)$.

Inklusion für konvexe Polytope als lineare Formulierung Ebenso wie die Kollisionsvermeidung für konvexe Polytope bei ausschließlicher Verwendung von translatorischen Freiheitsgraden linear abgebildet werden kann, kann auch die Inklusion als gemischt-ganzzahliges lineares System von Ungleichungen dargestellt werden. Dafür muss die Formulierung für das Kollisionsverbot nur geringfügig abgeändert werden, indem die Richtung der validen Positionierung hinsichtlich der Normalen der separierenden Ebenen umgekehrt wird und nicht mehr der Punkt getestet wird, welcher bezüglich der Ebene die maximal negative Entfernung einnimmt, sondern der bezüglich der Ebene die maximal positive Entfernung einnimmt. In anderen Worten ist die Inklusion sichergestellt, wenn der Eckpunkt v_j des Polytops Q, welcher bezüglich der f-ten Oberfläche des Polytops P die maximal positive Entfernung einnimmt, auf der negativen Seite der zugehörigen Ebene liegt, vorausgesetzt die Ebenennormale zeigt bei Polytopen immer nach außen.

Konkret bedeutet das, dass die $\leq$ Operatoren der Ungleichungen der linearen Kollisionsvermeidung gegen $\geq$ Operatoren ausgetauscht werden müssen und bei der Berechnung des Eckpunktes v_j statt dem Minimum das Maximum ausgewertet werden muss. Außerdem müssen nun nur noch die Oberflächen des umschließenden Polytops geprüft werden und die Bedingungen müssen nicht für mindestens einen Fall erzwungen werden, sondern gelten für alle umschließenden Flächen des Polytops P um das Polytop Q, was zur Folge hat, dass die Indikatorvariablen und die zugehörigen Big-M Konstanten nicht mehr berücksichtigt werden sollen. Dann lautet die zugehörige mathematische Darstellung zur Inklusion

$$-\sum_{d=1}^{D}\Big(n_{d,f} \cdot (\Delta_d^Q - \Delta_d^P)\Big) - h_f - h_f^Q \geq 0 \qquad f = 1, ..., F\ ,\ \ d = 1, ..., D \tag{4.138}$$

$$h_f^Q := max\Big\{\sum_{d=1}^{D}(n_{d,f} \cdot v_{d,j})\Big\} \qquad j = 1, ..., J\ ,\ \ f = 1, ..., F\ ,\ \ d = 1, ..., D \tag{4.139}$$

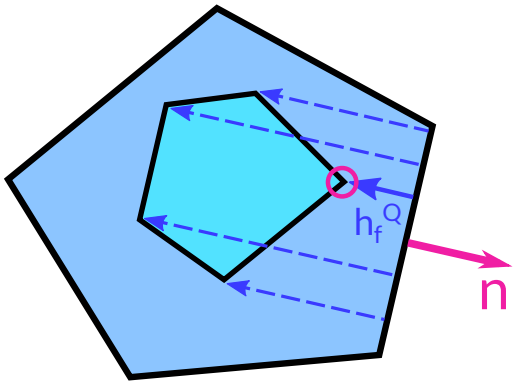


Abbildung 4.35: Inklusion für konvexe Polytope in 2D.

wobei die Formulierung zu F zusätzlichen Ungleichungen und keiner zusätzlichen Variable im Gesamtproblem führt. Die Formulierung ist aufgrund der ihr inhärenten Linearität in der Klasse `LinearContainment` implementiert und gilt ausschließlich für keine oder über Zustände berücksichtigte Skalierung und Rotation.

Inklusion für rechtwinklige Polytope als lineare Formulierung Für den Spezialfall der rechtwinkligen Polytope kann eine Formulierung verwendet werden, welche bei der Berücksichtigung von sowohl translatorischen als auch skalierenden Freiheitsgraden linearen Charakter hat. Diese ist in der `LinearContainment` Klasse implementiert und lautet

$$s_d^P \cdot (v_{d,i=0} + \lambda_d^P) + \Delta_d^P \leq s_d^Q \cdot (v_{d,j=0} + \lambda_d^Q) \qquad d = 1, ..., D \tag{4.140}$$

$$s_d^Q \cdot v_{d,j=0} + \Delta_d^Q \leq s_d^P \cdot v_{d,i=0} + \Delta_d^P \qquad d = 1, ..., D \tag{4.141}$$

wobei $v_{d,0}$ den linken unteren Eckpunkt des zugehörigen Polygons repräsentiert und λ_d die Länge des Rechtecks oder Quaders in die Raumrichtung d. Die Formulierung führt zu $2D$ zusätzlichen Ungleichungen und keinen zusätzlichen Variablen im Gesamtproblem. Auch hierbei dürfen die rotatorischen Freiheitsgrade nur in Form von Zuständen repräsentiert werden.

Mathematische Inklusionsformulierung für orbikulare Geometrien Für die Formulierung der Inklusion von orbikularen Geometrien können, ähnlich wie bei der Kollisionsvermeidung, die Mittelpunkte und Radien der Kreise oder Kugeln genutzt werden. Dabei umschließt eine orbikulare Geometrie P eine andere orbikulare Geometrie Q, wenn die Summe aus dem Abstand ihrer Mittelpunkte c^P und c^Q und dem Radius r^Q der umschlossenen orbikularen Geometrie kleiner ist als der Radius r^P der umschließenden orbikularen Geometrie. Dies äußert sich in mathematischer Form durch

$$\sum_{d=1}^{D} ((s^P \cdot c_d^P + \Delta_d^P) - (s^Q \cdot c_d^Q + \Delta_d^Q))^2 \leq (s^P \cdot r^P - s^Q \cdot r^Q)^2 \qquad d = 1, ..., D \tag{4.142}$$

mit den translatorischen Variablen Δ_d und den Skalierungsvariablen s. Aufgrund des quadratischen Charakters ist diese Formulierung in der `QuadraticContainment` Klasse zu finden. Sollen rotatorische Variablen berücksichtigt werden, kann dafür die `NonLinearContainment`

Klasse mit der folgenden Formulierung verwendet werden

$$\sum_{d=1}^{D}((s^P \cdot R_d^P \times c^P + \Delta_d^P) - (s^Q \cdot R_d^Q \times c^Q + \Delta_d^Q))^2 \leq (s^P \cdot r^P - s^Q \cdot r^Q)^2 \qquad d = 1, ..., D \tag{4.143}$$

mit $R_d = R_d(\rho_1, ..., \rho_D)$ als der d-ten Zeile der Rotationsmatrix $R(\rho_1, ..., \rho_D)$ mit den rotatorischen Variablen ρ_d für eine Rotation um die jeweilige Achse d.

Blackbox Inklusion Die in der `CodeContainment` Klasse umgesetzte Inklusion basiert auf der im *Design Cockpit 43*® [IILS, 2022] softwareinternen *Point-Inside-Mesh* Erkennung, welche bereits fester Bestandteil von Entwurfssprachen ist. Für die Inklusion wird ein System von Ungleichung aufgebaut, das für jeden Punkt dessen Enthaltensein prüft. Der Konditionalteil der Ungleichung ruft hierfür die *Point-Inside-Mesh* Methode auf. Gibt diese einen wahren booleschen Wert zurück, wird der Wert 1 verarbeitet, bei Nichtenthaltensein wird der Wert 0 verarbeitet. Für jeden Punkt wird gefordert, dass der Konditionalteil größer gleich 1 ist, was nur für ein Enthaltensein zutrifft. Diese auf Programmcode basierende Inklusionbedingung ist bei der Verwendung von deterministischen mathematischen Optimierungsverfahren nicht anwendbar. Es muss auf Optimierungsverfahren zurückgegriffen werden, welche in der Lage sind Blackbox Funktionen zu verarbeiten.

Nähe und Abstand

Neben dem Kollisionsverbot ist die Vorgabe von vorher definierten Abständen eine der am häufigsten benötigten Nebenbedingungen, die bei Packingprozessen berücksichtigt werden müssen. Im Ingenieurwesen sind sie bei der Berücksichtigung von Interferenzen in Verwendung, oder wenn unzulässige Gebiete für die Positionierung von Komponenten gemieden werden sollen, beispielsweise für eine ausreichende Wartungszugänglichkeit. In der Architektur und Stadtplanung sind sie essenziell für infrastrukturelle Problemstellungen, kommen aber bei der Optimierung von Belichtungs- und Komfortfragen zum Einsatz, wie z.B. der Definition einer minimalen Sichtweite nach außen. Sogar ästhetische Forderungen in beliebigen Produktentwürfen können über die Modellierung von Abständen mithilfe der ästhetischen Proportionierung optimiert werden. Im Folgenden sollen die im Laufe der vorliegenden Arbeit entwickelten und implementierten Bedingungen vorgestellt werden.

Abstand und Mindestabstand von Polytopen Um einen Mindestabstand zwischen allgemeinen konvexen Polytopen vorzugeben, können auch die Modelle für die Kollisionsvermeidung aus den `CollisionAvoidance` Klassen verwendet werden. Deren Formulierungen sind hierfür um den Abstand δ erweitert, welcher von dem/der Anwender/-in übergeben werden kann. Die allgemeine Formulierung aus *4.4.3 Mathematische Kollisionsvermeidung für konvexe Polytope* wird damit zu

$$\sum_{d=1}^{D}(n_d \cdot v_{d,i}) \geq u + 0.5 \cdot \delta \qquad \forall\ v_i \in P,\ i = 1, ..., I\ ,\quad d = 1, ..., D \tag{4.144}$$

$$\sum_{d=1}^{D}(n_d \cdot v_{d,j}) \leq u - 0.5 \cdot \delta \qquad \forall\ v_j \in Q,\ j = 1, ..., J\ ,\quad d = 1, ..., D \tag{4.145}$$

$$-1 \leq n_d \leq 1 \qquad d = 1, ..., D \tag{4.146}$$

$$\sum_{d=1}^{D} n_d^2 = 1 \qquad d = 1, ..., D \tag{4.147}$$

wobei aufgrund von δ keine zusätzliche Konstante ε mehr benötigt wird. Die zusätzliche Gleichung Gl. (4.147) ist für die Normierung der Hyperebene zuständig, da nur bei der normierten Darstellung der Hyperebene δ dem Abstand zwischen den Polytopen entspricht. In gleicher Weise sind auch die Ungleichungen in der `LinearCollisionAvoidance` Klasse, `QuadraticCollisionAvoidance` Klasse und `NonLinearCollisionAvoidance` Klasse erweitert worden. Gleiches gilt für die Formulierung aus *4.4.3 Lineare Kollisionsvermeidung für konvexe Polytope*, welche ebenso um den Abstand δ erweitert wurde. Eine zusätzliche Normierungsgleichung ist im linearen Fall allerdings nicht notwendig, da die Formulierung bereits normiert stattfindet. Soll anstatt eines Mindestabstandes ein vorher spezifizierter fester Abstand vorgegeben werden, müssen in allen Formulierungen die Operatoren $\leq$ und $\geq$ lediglich durch ein Gleichheitszeichen ersetzt werden. Diese Varianten sind in den jeweiligen `LinearDistance`, `QuadraticDistance` und `NonLinearDistance` Klassen implementiert.

Abstände von Punkten In der vorliegenden Arbeit sind zwei mathematische Darstellungen zur Abstandsberechnung implementiert, die Formulierung für die Manhattan-Distanz und die Formulierung für den euklidischen Abstand. Beide können sowohl als Randbedingung wie auch als Zielfunktion verwendet werden und werden daher in *4.4.4 Ziele für den Packingprozess* detailliert beschrieben.

Geometrischer Kontakt

Die optimale Lösung für eine Konfiguration von Produktkomponenten oder Modulen hängt essenziell von deren Schnittstellen untereinander ab. Materielle Schnittstellen setzen dabei häufig einen Kontakt der Komponentengeometrien voraus, wodurch der Lösungsraum eingeschränkt wird. Im Folgenden sollen die entwickelten Möglichkeiten der vorgestellten Entwurfssprache erörtert werden, mit denen Kontakbedingungen modelliert werden können.

Kontakt von Punkten Der Kontakt von Punkten kann einfach in mathematischer Form sichergestellt werden, indem die Gleichheit der Punktkoordinaten formuliert wird. Unter Berücksichtigung von ausschließlich translatorischen Freiheitsgraden und Skalierungsvariablen entspricht der Kontakt zweier Punkte p und q

$$s_d^P \cdot p_d + \Delta_d^P = s_d^Q \cdot q_d + \Delta_d^Q \quad d = 1, ..., D \tag{4.148}$$

und ist in der `LinearContact` Klasse implementiert. Für die Berücksichtigung von rotatorischen Variablen ist die Formulierung

$$s_d^P \cdot R_d^P \times p + \Delta_d^P = s_d^Q \cdot R_d^Q \times q + \Delta_d^Q \quad d = 1, ..., D \tag{4.149}$$

mit $R_d = R_d(\rho_1, ..., \rho_D)$ als der d-ten Zeile der Rotationsmatrix $R(\rho_1, ..., \rho_D)$ mit den rotatorischen Variablen ρ_d für eine Rotation um die jeweilige Achse d in der `NonLinearContact` Klasse implementiert.

Kontakt von orbikularen Geometrien Für den Kontakt von Kreisen und Kugeln werden die Formulierungen aus den `CollisionAvoidance` Klassen verwendet und deren minimal zulässiger Abstand als Kontakt interpretiert. Dann müssen lediglich die $\leq$ Operatoren gegen $=$ Operatoren ausgewechselt werden. Damit ergibt sich die in der `QuadraticContact` Klasse implementierte mathematische Darstellung

$$\sum_{d=1}^{D}((s^P \cdot c_d^P + \Delta_d^P) - (s^Q \cdot c_d^Q + \Delta_d^Q))^2 \geq (s^P \cdot r^P + s^Q \cdot r^Q)^2 \qquad d = 1, ..., D \tag{4.150}$$

und für den rotatorischen Fall die in der `NonLinearContact` Klasse implementierte Variante

$$\sum_{d=1}^{D}((s^P \cdot R_d^P \times c^P + \Delta_d^P) - (s^Q \cdot R_d^Q \times c^Q + \Delta_d^Q))^2 \geq (s^P \cdot r^P + s^Q \cdot r^Q)^2 \qquad d = 1, ..., D \tag{4.151}$$

Kontakt von rechtwinkligen Polytopen Der Kontakt von rechtwinkligen Polytopen spielt unter anderem in der Layoutplanung wie z.B. in der Architektur eine Rolle. Notwendige Kontakte ergeben sich dabei häufig aus den Topologien von Grundrissen und der Vernetzung der enthaltenen Räume. Neben dem bloßen Kontakt spielt dabei auch die Kontaktbreite oder -fläche eine Rolle. Ein konkretes Beispiel wäre hierfür, dass Durchgänge zwischen Räumlichkeiten sinnvolle Maße annehmen müssen. In vielen solcher Fälle müssen dabei keine Rotationen betrachtet werden, sodass die Positionierung an den Raumachsen ausgerichtet werden kann. Für genau diesen Anwendungsfall wurde in der vorliegenden Arbeit eine Kontaktrepräsentation speziell für rechtwinklige Polytope modelliert, welche neben der reinen Kontaktforderung auch die Vorgabe von minimalen Kontaktmaßen zulässt. Die zugehörige, in der `LinearContact` Klasse implementierte, mathematische Formulierung wurde auf Basis der Formulierung zur Kollisionsvermeidung aus *4.4.3 Lineare Kollisionsvermeidung für rechtwinklige Polytope* entwickelt, mit deren Hilfe bereits die gegenseitige Positionierung in einer Raumrichtung repräsentiert werden kann. Ein Kontakt setzt aber immer zusätzliche Bedingungen in den übrigen Raumrichtungen voraus. Es soll zunächst die gegenseitige Positionierung zweier rechtwinkliger Polytope P und Q betrachtet werden. Im Gegensatz zur Kollisionsvermeidung muss der Kontakt erzwungen werden, das heißt, dass die Ungleichungen zu Gleichungen werden müssen. Es muss also für die Raumachse $d = d_n$ normal zur Kontaktebene gelten

$$s_d^P \cdot (v_{d,i=0} + \lambda_d^P) + \Delta_d^P = s_d^Q \cdot v_{d,j=0} + \Delta_d^Q \qquad d = d_n \tag{4.152}$$

$$\text{oder} \qquad s_d^Q \cdot (v_{d,j=0} + \lambda_d^Q) + \Delta_d^Q = s_d^P \cdot v_{d,i=0} + \Delta_d^P \qquad d = d_n \tag{4.153}$$

wobei $v_{d,0}$ den linken unteren Eckpunkt des zugehörigen Polytops repräsentiert und λ_d die Länge in die Raumrichtung d. Diese Ungleichungen stellen in einem ersten Schritt sicher, dass jeweils zwei Flächen der Polytope entgegengesetzt in einer Ebene liegen. Nun muss noch sichergestellt werden, dass sich die angrenzenden und begrenzten Flächen der Polytope tatsächlich berühren und nur in einem zulässigen Maß, sowohl in positive wie auch in negative Richtung, gegeneinander verschoben sind. Diese Bedingung muss für alle weiteren, die Kontaktebene aufspannenden, Raumrichtungen gelten. Für die untere und obere Begrenzung können pro Raumrichtung die beiden Bedingungen

$$s_d^P \cdot (v_{d,i=0} + \lambda_d^P) + \Delta_d^P \geq s_d^Q \cdot v_{d,j=0} + \Delta_d^Q + \mu_d \qquad d = 1, ..., D \ , \ d \neq d_n \tag{4.154}$$

$$s_d^P \cdot v_{d,i=0} + \Delta_d^P \leq s_d^Q \cdot (v_{d,j=0} + \lambda_d^Q) + \Delta_d^Q - \mu_d \qquad d = 1, ..., D \ , \ d \neq d_n \tag{4.155}$$

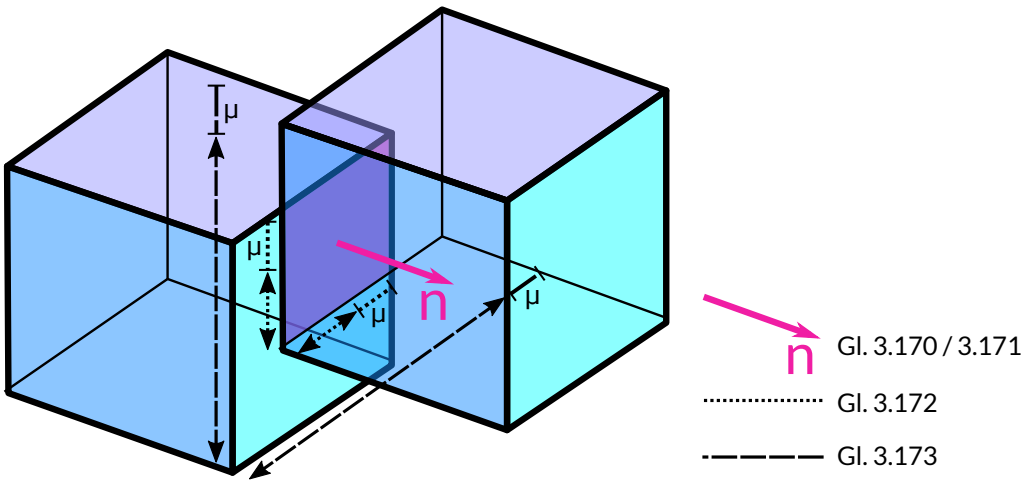


Abbildung 4.36: Kontakt von rechtwinkligen achsenorientierten Polytopen in 3D.

mit der Überlappungslänge μ_d pro Raumrichtung d aufgestellt werden, wobei alle Ungleichungspaare aktiv sein müssen, für die gilt $d \neq d_n$. Abbildung 4.36 stellt diesen Sachverhalt am Beispiel zweier Quader dar.

Die gegenseitige Abhängigkeit kann mithilfe der bereits vorgestellten Szenarien modelliert werden. Es ergibt sich für $d = 1, ..., D$ das folgende Gesamtproblem mit den booleschen Variablen Θ^{PQ}, Θ^{QP} und Θ_d und einer ausreichend großen Konstante M

$$s_d^P \cdot (v_{d,i=0} + \lambda_d^P) + \Delta_d^P \leq s_d^Q \cdot v_{d,j=0} + \Delta_d^Q + M \cdot (2 - \Theta^{PQ} - \Theta_{d_n}) \qquad d \neq d_n \quad (4.156)$$

$$s_d^P \cdot (v_{d,i=0} + \lambda_d^P) + \Delta_d^P \geq s_d^Q \cdot v_{d,j=0} + \Delta_d^Q + M \cdot (2 - \Theta^{PQ} - \Theta_{d_n}) \qquad d \neq d_n \quad (4.157)$$

$$s_d^Q \cdot (v_{d,j=0} + \lambda_d^Q) + \Delta_d^Q \leq s_d^P \cdot v_{d,i=0} + \Delta_d^P + M \cdot (2 - \Theta^{QP} - \Theta_{d_n}) \qquad d \neq d_n \quad (4.158)$$

$$s_d^Q \cdot (v_{d,j=0} + \lambda_d^Q) + \Delta_d^Q \geq s_d^P \cdot v_{d,i=0} + \Delta_d^P + M \cdot (2 - \Theta^{QP} - \Theta_{d_n}) \qquad d \neq d_n \quad (4.159)$$

$$s_{d_n}^P \cdot (v_{d_n,i=0} + \lambda_{d_n}^P) + \Delta_{d_n}^P \geq s_{d_n}^Q \cdot v_{d_n,j=0} + \Delta_{d_n}^Q + \mu_{d_n} - M \cdot \Theta_{d_n} \qquad d_n = 1, ..., D \quad (4.160)$$

$$s_{d_n}^P \cdot v_{d_n,i=0} + \Delta_{d_n}^P \leq s_{d_n}^Q \cdot (v_{d_n,j=0} + \lambda_{d_n}^Q) + \Delta_{d_n}^Q - \mu_{d_n} + M \cdot \Theta_{d_n} \qquad d_n = 1, ..., D \quad (4.161)$$

$$\Theta^{PQ} + \Theta^{QP} = 1 \quad (4.162)$$

$$\sum_{d=1}^{D} \Theta_d = 1 \qquad d = 1, ..., D \quad (4.163)$$

wobei über die Indikatorvariablen Θ_d die Richtung der Kontaktebenen definiert ist und die Indikatorvariablen Θ^{PQ} und Θ^{QP} angeben auf welcher der beiden Seiten der rechtwinkligen Polytope sich der Kontakt befindet. Die Ungleichungen Gl. (4.156) bis Gl. (4.159) ersetzen die Gleichungen Gl. (4.152) und Gl. (4.153) unter Berücksichtigung der Indikatorvariablen. Die Formulierung ergibt $(6 \cdot D + 2)$ zusätzliche Ungleichungen und $(2 + D)$ zusätzliche boolesche Variablen für das Gesamtproblem, wenn keine *kompakte* Repräsentation (siehe *4.3.4 Effiziente Modellierung von Szenarien*) gewählt wird.

Packingunabhängige Nebenbedingung

Das vorgestellte Packingframework stellt neben den auf den Packingprozess zugeschnittenen Nebenbedingungen auch weitere Helferklassen für allgemeine Randbedingungen zur Verfügung. Hierzu gehören beispielsweise die Auswahl eines minimalen und die Auswahl eines maximalen Wertes aus einer Liste von Entwurfsparametern. Diese im Folgenden als MIN- und MAX-Funktion bezeichneten Ausdrücke sind nicht zwingend dem Packingprozess zuzuordnen und werden daher in die Kategorie der allgemeinen Optimierungsbedingungen eingeordnet.

Minima als Nebenbedingung Die Aushwahl eines Minimums m aus einer vorher definierten Reihe an möglichen Werten oder Variablen wird über die `MinBuilder` Klasse zur Verfügung gestellt. Die mathematische Formulierung lautet dabei

$$
\begin{aligned}
m = \text{MIN}\{m_0, ..., m_I\} \quad \Rightarrow \quad & m \leq m_i && i = 0, ..., I && (4.164)\\
& m \geq m_i - M \cdot (1 - \Theta_i) && i = 0, ..., I && (4.165)\\
& \sum_{i=1}^{I} \Theta_i = 1 && i = 0, ..., I && (4.166)
\end{aligned}
$$

mit einer zusätzlichen Ersatzvariable m, einer ausreichend großen Konstante M und den Indikatorvariablen Θ_i. Ist die Anzahl der zur Auswahl stehenden Werte gerade, kann die Summengleichung Gl. (4.166) vernachlässigt werden, wenn eine *kompakte* Modellierung der Indikatorvariablen stattfindet. Ist die MIN-Funktion Teil einer Maximierungszielfunktion, können sowohl Gl. (4.166) als auch Gl. (4.165) vernachlässigt werden, da durch die übrigen Ungleichungen der minimale Listenwert der für die Maximierung kritische ist und dadurch der Variable m automatisch der korrekte minimale Wert zugewiesen wird.

Maxima als Nebenbedingung Auf dieselbe Weise kann auch eine MAX-Funktion als mathematisches System von Ungleichungen ausgedrückt werden. Die zugehörige Formulierung ist in der `MaxBuilder` Klasse implementiert und kann dargestellt werden als

$$
\begin{aligned}
m = \text{MAX}\{m_0, ..., m_I\} \quad \Rightarrow \quad & m \geq m_i && i = 0, ..., I && (4.167)\\
& m \leq m_i + M \cdot (1 - \Theta_i) && i = 0, ..., I && (4.168)\\
& \sum_{i=1}^{I} \Theta_i = 1 && i = 0, ..., I && (4.169)
\end{aligned}
$$

mit einer zusätzlichen Ersatzvariable m, einer ausreichend großen Konstante M und den Indikatorvariablen Θ_i. Auch hier kann Gl. (4.169) vernachlässigt werden, wenn eine *kompakte* Modellierung möglich ist. Außerdem können ebenfalls Gl. (4.169) als auch Gl. (4.168) vernachlässigt werden, wenn die MAX-Funktion Teil einer Minimierungszielfunktion ist, da die Ungleichung mit dem maximalen Listenwert die kritische Grenze darstellt, sodass die Variable automatisch den maximalen Wert annimmt.

Auswahl an Nebenbedingungen Nicht immer sind alle modellierten Restriktionen gleichzeitig notwendig. Bei der Modellierung von Nebenbedingungen können auch Fälle auftreten, in denen lediglich eine aus einem Satz an Bedingungen eingehalten werden muss. Für diesen Fall ist die Klasse `OrBuilder` entwickelt worden. An diese kann eine ganze Reihe an Bedingungen übergeben werden, aus welchen mindestens eine Bedingung erfüllt sein muss. Die Klasse führt für jede der I Bedingungen eine Indiaktorvariable Θ_i ein und erweitert deren mathematische

Formulierung mit dieser und einer ausreichend großen Konstante M zur Inaktivierung bzw. globalen Validität. Es ergeben sich dabei die Änderungen und Erweiterungen

$$condition_i \ \leq / \geq \ rhs_i \ +/- \ M \cdot (1 - \Theta_i) \qquad i = 0, ..., I \tag{4.170}$$

$$\sum_{i=1}^{I} \Theta_i \geq 1 \qquad i = 0, ..., I \tag{4.171}$$

wobei Gl. (4.171) sicherstellt, dass mindestens eine der Bedingungen erfüllt ist.

Spezialfälle

Bei der Modellierung von Packingproblemen treten weitere Spezialfälle auf, welche grundsätzlich alle bereits erläuterten Packingbedingungen betreffen und sich somit auf deren mathematische Darstellung auswirken können. Diese sind in zwei Hauptkategorien aufgeteilt, welche im Folgenden erläutert werden sollen.

Nichtkonvexe Geometrien und Verbundgeometrien Sollen Nebenbedingungen für nichtkonvexe Geometrien aufgestellt werden, kann der/die Anwender/-in zwischen zwei Optionen wählen. Die erste ist die Modellierung der Zwänge in Form von Programmcode unter Zuhilfenahme der `CodeConstraintBuilder` Klassen. Als Folge kann bei dieser Variante kein deterministisches mathematisches Verfahren für die Optimierung verwendet werden. Sollen alle Bedingungen in mathematischer Form vorliegen, müssen die nichtkonvexen Geometrien in konvexe Teilgeometrien aufgespalten werden. Dafür stehen zahlreiche Vereinfachungen aus der Ontologie der Entwurfssprache für Geometrieapproximationen (siehe Abschnitt 4.2.2) zur Verfügung. Geometrien, welche aus Subgeometrien bestehen, werden in der Entwurfssprache als `GDCompound` Objekte verarbeitet. Alle Bedingungen müssen dann für alle zugehörigen `GDSingle` Objekte separat modelliert werden. Für den Fall der Kollisionsvermeidung ist der Umgang mit Verbundgeometrien bereits in den `ConstraintBuilder` Klassen integriert und wird zur Laufzeit umgesetzt, wenn `GDCompound` Objekte übergeben wurden. Im Fall der Inklusion ist in der vorliegenden Arbeit der einfachere Fall abgedeckt, in dem eine konvexe Geometrie einen Verbund aus konvexen Geometrien umschließt. Bei diesem müssen lediglich die Formulierungen für alle Teilgeometrien in das Gesamtproblem integriert werden. Das ist leicht nachvollziehbar, denn wenn eine konvexe Geometrie einen Verbund aus konvexen Geometrien umschließt, umschließt sie gleichzeitig auch jede einzelne der Verbundgeometrien. Der schwierigere Fall, in dem ein Verbund aus konvexen Geometrien eine einzelne inkludiert, wurde für keinen der konkreten Anwendungsfälle benötigt und daher vorerst vernachlässigt.

Die Verwendung von Szenarien Sollen Freiheitsgrade zugunsten von einfacheren mathematischen Darstellungen in Form von Zuständen verwendet werden, müssen auch diese bei der Formulierung des Gesamtproblems berücksichtigt werden. Hierfür müssen alle gewünschten Nebenbedingungen für jeden einzelnen möglichen Zustand (als Kombination von Freiheitsgradzuständen) modelliert und sinnvoll zusammengesetzt werden. Die in der vorliegenden Arbeit entwickelten `ConstraintsBuilder` berücksichtigten dies automatisch und berechnen über eine Iteration die mathematischen Systeme für alle modellierten Zustände der eingehenden `PackagingComponent` Objekte und weisen diese den zugehörigen Indikatorvariablen zu. Dabei wird sichergestellt, dass die zu einem Zustand gehörenden Bedingungen genau dann aktiv sind, wenn auch der Zustand selbst aktiv ist.

4.4.4 Ziele für den Packingprozess

Neben einer Bibliothek für Nebenbedingungen im Packingprozess wird in der Entwurfssprache auch eine Sammlung an vordefinierten packingsspezifischen Zielfunktionen zur Verfügung gestellt. Ebenso wie die Packingebedingungen, müssen auch die Ziele durch ein elementares `FitnessFunction` Objekt und ein System von elementaren `Constraint` Objekten repräsentiert werden. Abb. 4.37 gibt einen Überblick über die entwickelte Packingziele, wobei einige auch in Form von Nebenbedingungen im Framework integriert sind, worauf aufgrund der Ähnlichkeit nicht explizit im vorangegangenen Kapitel eingegangen wurde. Ein generelles Packingziel ist in Form eines `PackingObjectiveBuilder` Interfaces umgesetzt, welches eine Optimierungszielfunktion als Variable hält. Die Repräsentation der Zielfunktionen findet dabei ausschließlich in linearer Form statt. Um dies zu gewährleisten, werden nichtlineare Zielfunktionen derartig umformuliert, dass die Zielfunktion linear vorliegt und der nichtlineare Anteil der ursprünglichen Zielformulierung als System von Ungleichungen repräsentiert wird. Dies ist durch die Einführung einer Ersatzvariable und einer zusätzlichen Ungleichung grundsätzlich immer möglich. Der Grund für diese Art der Darstellung liegt in ihrer Flexibilität für die spätere Optimierung, da lineare Zielfunktionen von jedem der angebundenen Optimierer interpretiert und verarbeitet werden können. Es sei also an dieser Stelle bereits darauf hingewiesen, dass alle im weiteren Verlauf der vorliegenden Arbeit vorgestellten Zielfunktionen in linearer Form formuliert sind. Hinsichtlich der zugehörigen Ungleichungen folgt die Grundstruktur der Packingziele derselben Hierarchie wie die Ontologie der Nebenbedingungen und Zwänge. Dies wird in Abb. 4.37 über die importierten `LinearConstraintBuilder`, `QuadraticConstraintBuilder`, `NonLinearConstraintBuilder` und `CodeConstraintBuilder` Klassen ersichtlich, welche ihre grundlegenden Charakteristiken an die einzelnen Packingzielklassen vererben. Im Folgenden sollen die implementierten Zielmodelle detailliert vorgestellt werden.

Optimierung der Abstände zueinander oder zu anderen Gebieten

Im Ingenieurwesen müssen beim Auffinden von optimalen Bauteilkonfigurationen nicht selten bestimmte vorgegebene Abstände optimiert werden. In vielen Fällen wird dabei eine Minimierung der Abstände zwischen verknüpften Bauteilen gefordert, beispielsweise um das Verkabelungs- oder Verrohrungsgewicht zu reduzieren. Andere Anwendungsfälle lassen sich in der Architektur und Stadtplanung beobachten, in denen beispielsweise die Verbindungswege zwischen einzelnen Standorten minimiert werden sollen, um eine möglichst effiziente Vernetzung zu erreichen. Auch die Maximierung des Abstands kann ein wichtiger Zielparameter sein. Ein Beispiel aus dem Alltag, welches dem Leser sicherlich schon allzu häufig begegnet ist, ist im Bereich der Flugzeugkabinenplanung zu finden. Dort beeinträchtigt der Sitzabstand zu einem Großteil den Sitzkomfort, daher wäre dessen Maximierung durchaus wünschenswert, würde man den Fluggast fragen. Ein weiterer allgemeiner Fall im Ingenieurwesen ist beispielsweise die Maximierung des Abstandes von bestimmten Komponenten zu einer Störquelle. Das Pendant in der Architektur und Stadtplanung wäre dann zum Beispiel die Maximierung des Abstandes von einem Standort zu einer möglichen Lärmquelle. Für alle solche und ähnliche Fälle sind im Laufe der vorliegenden Arbeit Abstandszielfunktionen in das vorgestellte Framework integriert worden, welche im Folgenden vorgestellt werden sollen.

Lineare Formulierung der Manhattan-Distanz Soll die Distanz zwischen zwei Punkten gemessen werden, ist der intuitive Ansatz die Berechnung des natürlichen Abstands, welcher auch als euklidischer Abstand bezeichnet wird. Allerdings ist nicht immer der euklidische Abstand sinnvoll. Gerade im Bereich der Architektur und Stadtplanung kann diese sog. Luftlinie nicht

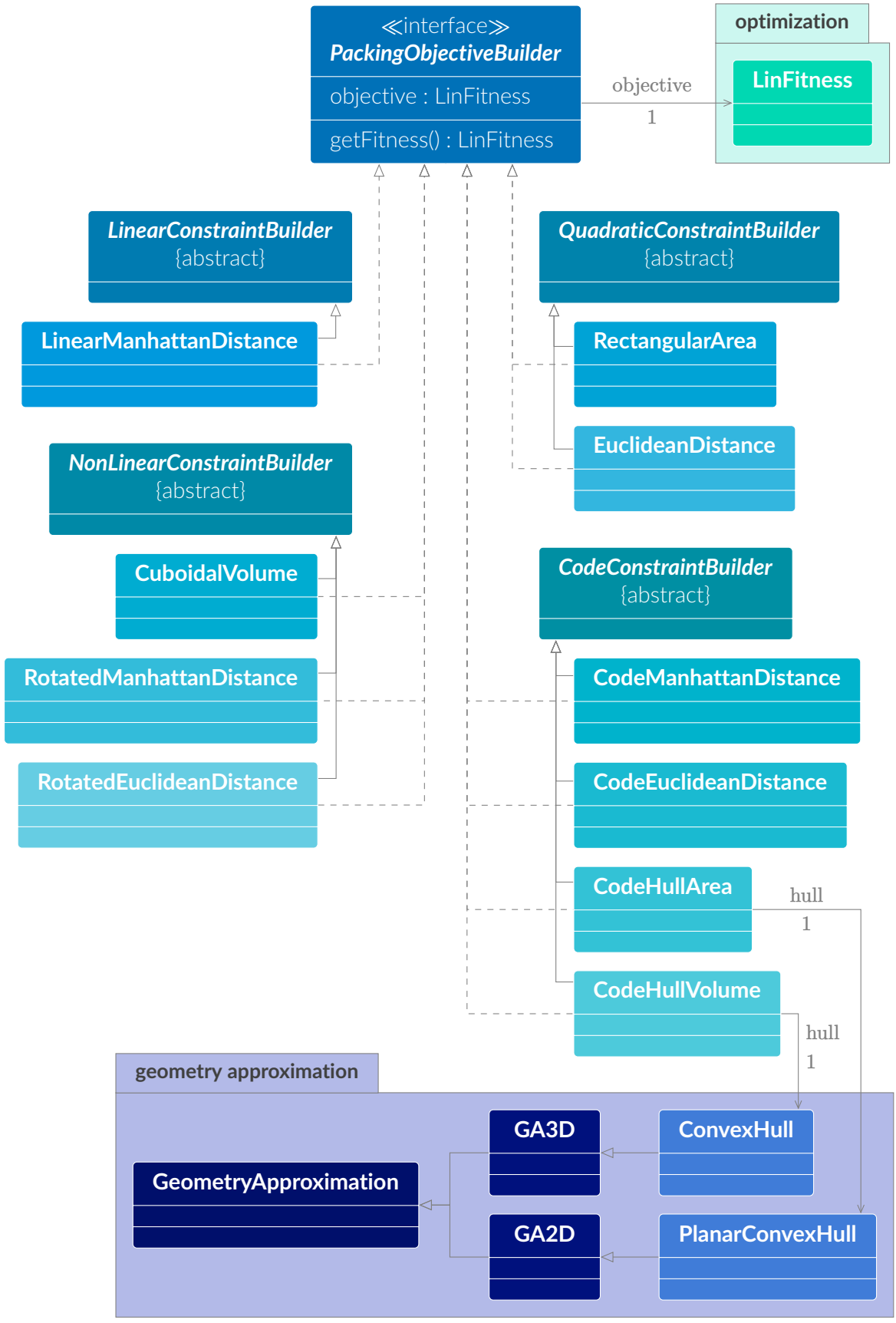


Abbildung 4.37: Ontologie für packingspezifische Ziele mit importierten Klassen aus der Ontologie für Geometrieapproximation (dunkelblau) und der Optimierungsontologie (grün).

am Boden überwunden werden, da viele Hindernisse den Weg versperren, welche umgangen werden müssen. Als Alternative zum euklidischen Abstand kann die sog. Manhattanmetrik oder Manhattan-Distanz verwendet werden, in welcher die Distanz m zwischen zwei Punkten p und q durch die Summe der Abstände ihrer Einzelkoordinaten in die Raumrichtungen d repräsentiert wird.

$$m = \sum_{d=1}^{D} |p_d - q_d| \qquad d = 1, ..., D \tag{4.172}$$

In Übereinstimmung mit zu der zur Erläuterung gewählten Domäne der Stadtplanung hat die Bezeichnung der Metrik ihren Ursprung in Manhattans orthogonalem Straßennetz, in dem ein zurückzulegender Weg durch eine Aneinanderreihung von vertikalen und horizontalen Wegstücken berechnet werden kann. Für die mathematische Formulierung in der vorliegenden Arbeit kann die obige Darstellung um die translatorischen Freiheitsgrade s_d und Skalierungsvariablen Δ_d erweitert werden zu

$$min/max \sum_{d=1}^{D} |(s_d^P \cdot p_d + \Delta_d^P) - (s_d^Q \cdot q_d + \Delta_d^Q)| \qquad d = 1, ..., D \tag{4.173}$$

und so als Manhattan-Distanz verwendet werden. Der Betrag soll im folgenden Abschnitt separat besprochen werden. Es wird an dieser Stelle allerdings bereits vorweggegriffen, dass die Formulierung des Betrags linear dargestellt werden kann, weshalb auch die Manhattan-Distanz als lineare Formulierung repräsentiert werden kann. Voraussetzung ist allerdings die Darstellung von möglichen rotatorischen Freiheitsgraden über Zustände. Dann kann für die Anwendung der Manhattan-Distanz als Optimierungsparameter die `LinearManhattanDistance` Klasse verwendet werden. Soll die Rotation als Variable in das Optimierungsproblem eingehen, verändert sich die Formulierung zu

$$min/max\ m \tag{4.174}$$

$$m\ \geq / \leq \sum_{d=1}^{D} |(s_d^P \cdot R_d^P \times p + \Delta_d^P) - (s_d^Q \cdot R_d^Q \times q + \Delta_d^Q)| \qquad d = 1, ..., D \tag{4.175}$$

mit $R_d = R_d(\rho_1, ..., \rho_D)$ als der d-ten Zeile der Rotationsmatrix $R(\rho_1, ..., \rho_D)$ mit den rotatorischen Variablen ρ_d für eine Rotation um die jeweilige Achse d. Die Formulierung ist in dieser Form in der `RotatedManhattanDistance` Klasse implementiert.

Lineare Formulierung des Betrags Wie in vorigem Abschnitt angedeutet, muss auch die Darstellung des Betrags im vorgestellten Framework abgebildet werden. Zu diesem Zweck wurde die `AbsBuilder` Klasse entwickelt. Für die Formulierung des Betrags wird eine Ersatzvariable eingeführt, welche den gesamten Ausdruck ersetzten soll. Nimmt man die Manhattan-Distanz und eine Minimierung als Beispiel, gilt also

$$min\ |p_x - q_x| \ \Rightarrow\ min\ x \tag{4.176}$$

wobei die neu eingeführte Variable x sinnvoll beschränkt werden muss, sodass gleichzeitig auch $|p_x - q_x|$ minimiert wird. Werden die möglichen Ergebnisse, die aus einem Betrag resultieren, genauer betrachtet, kann festgestellt werden, dass es eigentlich nur die zwei Varianten

$$|p_x - q_x| = \begin{cases} p_x - q_x & \text{wenn } p_x \geq q_x \\ -p_x + q_x & \text{sonst} \end{cases} \tag{4.177}$$

gibt. Für ein Wertepaar ergibt sich innerhalb des Betrags immer ein positiver und ein negativer Wert, wovon per Definition immer der positive betrachtet werden muss. Aus dieser Erkenntnis ergeben sich zusätzliche Beschränkungen und die Minimierung kann dargestellt werden als

$$min\ x \tag{4.178}$$
$$p_x - q_x \leq x \tag{4.179}$$
$$-p_x + q_x \leq x \tag{4.180}$$

wobei wie gewünscht automatisch der positive Wert die Minimierung nach unten begrenzt, da er die höhere der beiden Grenzen darstellt. Es soll nun der Maximierungsfall betrachtet werden. Bei der Maximierung einer Ersatzvariable muss sichergestellt werden, dass diese immer kleiner oder gleich dem ursprünglich zu maximierenden Term ist, sonst würde die Ersatzvariable unabhängig von diesem maximiert werden. Die zur Minimierung analogen Ungleichungen ergeben sich zu $x \leq p_x - q_x$ und $x \leq -p_x + q_x$, wodurch die Maximierung unnötig stark eingeschränkt wird. Da beide Schranken gleichzeitig aktiv sind, schränkt die untere der beiden Grenzen die Maximierung ein, was auch bedeutet, dass immer der negative Wert statt des gewünschten positiven Wertes als Ersatz für den Betrag verwendet wird. Für die Deaktivierung der „negativen“ Begrenzung, muss die Fallunterscheidung

$$max\ x \tag{4.181}$$
$$x \leq p_x - q_x + M \cdot \Theta \tag{4.182}$$
$$x \leq -p_x + q_x + M \cdot (1 - \Theta) \tag{4.183}$$

eingeführt werden, mit einer ausreichend großen Konstante M und einer zusätzlichen booleschen Variable Θ. Soll der Betrag nicht in einer Zielfunktion, sondern in einer Nebenbedingung verwendet werden, gilt die Formulierung für die Minimierung, wenn der Betrag durch die Ungleichung nach oben beschränkt ist und die Formulierung für die Maximierung, wenn der Betrag nach unten beschränkt ist. Bei der vorgestellten Betrags-Repräsentation wird der mathematische Charakter des Ausdrucks innerhalb des Betrags nicht verändert, d.h. der Grad des Ausdrucks entspricht immer auch dem Grad der Betragsformulierung.

Quadratische Formulierung des euklidischen Abstands Die Formulierung des euklidischen Abstands e kann über dessen quadratische Darstellung erfolgen. Dabei hängt die Darstellung von der Zielsetzung der Zielfunktion ab, oder in anderen Worten davon, ob eine Minimierung oder Maximierung stattfinden soll. Es soll zunächst der quadratische Fall ohne rotatorische Variablen betrachtet werden, welcher in der `EuclideanDistance` Klasse implementiert ist. Für eine Minimierung oder Maximierung gilt dann

$$min/max\ \sum_{d=1}^{D}((s_d^P \cdot p_d + \Delta_d^P) - (s_d^Q \cdot q_d + \Delta_d^Q))^2 \qquad d = 1, ..., D \tag{4.184}$$
$$\tag{4.185}$$

Spielt der tatsächliche euklidische Abstand eine Rolle, beispielsweise wenn der Abstand als einzelnes Kriterium einer multikriteriellen Zielfunktion eingeht, müssen eine zusätzliche Variable $e_{squared}$ und eine zusätzliche Ungleichung eingeführt werden. Dann gilt für die Minimierung

$$min\ e \tag{4.186}$$
$$e_{squared} \geq \sum_{d=1}^{D}((s_d^P \cdot p_d + \Delta_d^P) - (s_d^Q \cdot q_d + \Delta_d^Q))^2 \qquad d = 1, ..., D \tag{4.187}$$
$$e^2 \geq e_{squared} \tag{4.188}$$

und für die Maximierung

$$max\ e \tag{4.189}$$

$$e_{squared} \leq \sum_{d=1}^{D}((s_d^P \cdot p_d + \Delta_d^P) - (s_d^Q \cdot q_d + \Delta_d^Q))^2 \qquad d = 1, ..., D \tag{4.190}$$

$$e^2 \leq e_{squared} \tag{4.191}$$

Bei der Verwendung als Nebenbedingung kann die Formulierung für die Minimierung angewandt werden, wenn die Nebenbedingung nach oben beschränkt ist und die Formulierung für die Maximierung, wenn die Nebenbedingung nach unten beschränkt ist. Die Formulierung für die Berücksichtigung der rotatorischen Freiheitsgrade als Variable kann durch Einbeziehung der Rotationsmatrix $R_d = R_d(\rho_1, ..., \rho_D)$ gebildet werden und ist in dieser Form in der `RotatedEuclideanDistance` Klasse implementiert.

Optimierung des Flächeninhaltes oder Volumens

Rechtwinklige Flächen und Volumina Eine besonders häufig in der Layout- oder Konfigurationsoptimierung auftretende Zielgröße ist die Gesamtfläche, welche alle zu platzierenden Elemente beinhaltet, oder das alle Bauteile umschließende Gesamtvolumen. Für die Umsetzung der Anwendungsbeispiele aus Kapitel 5 wurden in der vorliegenden Arbeit Optimierungsrepräsentationen für rechtwinklige Flächen und Volumina implementiert. Die zugehörigen Klassen sind je nach gewählten Freiheitsgraden die `RectangularArea` und `CubicVolume` Klassen aus Abb. 4.37. Dabei haben die Zieldarstellungen genau zwei Aufgaben. Zum einen die (triviale) Repräsentation einer Rechtecksfläche P mit $D = 2$ bzw. eines Quadervolumens P mit $D = 3$ als

$$min/max\ P \tag{4.192}$$

$$P \geq / \leq \prod_{d}^{D}(s_d^P \cdot \lambda_d^P) \qquad d = 1, ..., D \tag{4.193}$$

wobei λ_d die Länge des Rechtecks oder Quaders in die Raumrichtung d repräsentiert und s_d die Skalierung um den Faktor s in die Raumrichtung d. Die zweite Aufgabe betrifft die (nichttriviale) Sicherstellung, dass alle zu positionierenden Komponenten zuverlässig umschlossen werden. Letzteres kann über die Inklusionsformulierungen erreicht werden, welche in Abschnitt 4.4.3 nachzulesen sind. Diese gehen neben der eigentlichen linearen Zielfunktion in Form von zusätzlichen Nebenbedingungen in die jeweilige Zielrepräsentation mit ein. Für rotationsbehaftete Fälle sind aufgrund der zum Zeitpunkt der Erstellung der vorliegenden Arbeit fehlenden Notwendigkeit keine mathematischen umhüllenden Flächen- oder Volumenmodelle implementiert. Für diese Fälle können die in Programmcode formulierten Klassen verwendet werden.

Flächen und Volumina von Polytopen Für eine genauere Einschätzung der umschließenden Fläche oder des umhüllenden Volumens können die Klassen `CodeHullArea` und `CodeHullVolume` verwendet werden. Diese basieren auf den Geometrieapproximationen `PlanarConvexHull` (sieheAbschnitt 4.2.2) und `ConvexHull` (sieheAbschnitt 4.2.2) aus der bereits zu Anfang vorgestellten Entwurfssprache für Geometrieapproximationen. Dabei werden alle Eckpunkte aus den zu platzierenden Bauteilen extrahiert und es werden die umhüllenden konvexen Hüllen berechnet. Der Flächeninhalt oder das Volumen der konvexen Polytope kann in jedem Zeitschritt unter Zuhilfenahme der in Entwurfssprachen integrierten *VTK* Bibliothek [Schroeder et al., 2006] berechnet werden.

Multikriterielle Zielfunktionen

Bei realen Layout- oder Packingproblemen ist es mitunter schwer bis unmöglich, ein einziges Ziel bzw. einen einzelnen Bewertungsparameter zu benennen, welcher optimiert werden müsste. Viel mehr müssen, wie im Ingenieurwesen üblich, verschiedene Entwurfskriterien gleichzeitig optimiert werden, was zu der sog. multikriteriellen Optimierung führt. Die vorgestellte Entwurfssprache sieht für diesen Fall eine Summation aller Bewertungskriterien vor, was durch die `ObjectiveAggregator` Klasse unterstützt wird. Um die unterschiedlichen Bedeutungen der einzelnen Bewertungskriterien allerdings zu berücksichtigen, kann eine Gewichtung der einzelnen Zielparameter vorgenommen werden. Dann ergibt sich eine multikriterielle Zielfunktion oder zu engl. Fitness f_a für eine Alternative a als

$$f_a = \sum_{c=1}^{C} w_c \cdot \kappa_a \qquad a = 1, ..., A \ , \ c = 1, ..., C \tag{4.194}$$

mit der Gewichtung w_c des zugehörigen ausgewerteten Kriteriums κ_a für die Alternative a mit insgesamt $c = 1, ..., C$ Kriterien (von engl. *criterion*). Gehen Kriterien mit unterschiedlichen Zielen ein, oder in anderen Worten, sollen gewisse Parameter minimiert und andere maximiert werden, muss die Zusammensetzung so erfolgen, dass die Gesamtzielfunktion minimiert wird und alle zu maximierenden Parameter als Subtrahend in die Summe eingehen, woraus sich

$$f_a = \sum_{c=1}^{C_{min}} w_c \cdot \kappa_a^{min} - \sum_{c=C_{min}+1}^{C_{max}} w_c \cdot \kappa_a^{max} \qquad a = 1, ..., A \ , \ c = 1, ..., C_{min}, ..., C_{max} \tag{4.195}$$

ergibt. Diese Repräsentation ist einfach umzusetzen, es sei aber darauf hingewiesen, dass eine Vielzahl anderer Bewertungsmethoden existiert, welche genauso gut oder besser sein können. Die Auswahl einer geeigneten Bewertung stellt allerdings einen eigenen Forschungsbereich dar, eine detaillierte Auseinandersetzung würde den Rahmen der vorliegenden Arbeit sprengen und wird daher nicht geführt.

Zielfunktionen bei Verwendung von States

Sollen Parameter in der Zielfunktion berücksichtigt werden, welche über Zustände definiert sind, muss dafür Sorge getragen werden, dass nur diejenigen Parameter tatsächlich optimiert werden, welche für die Packingauslegung aktiv sind. Zunächst wird in einem solchen Fall eine Zielfunktion für jeden der möglichen Zustände gebildet. Zur Berücksichtigung von allen potenziellen Zuständen wird anschließend eine Summe aus den zustandsäbhängigen Zielfunktionen erstellt. Aus dieser Summe darf dann nur ein einziger Parameter nicht 0 sein. Dies kann durch ein System von Ungleichungen umgesetzt werden. Eine Maximierung kann mit den Ersatzvariablen $f_{state,\sigma}$ für eine Zielfunktion $F_{state,\sigma} \cdot \vec{v}_{state,\sigma}$ und den jeweiligen Zustand σ also repräsentiert werden als

$$max \sum_{\sigma=1}^{\Sigma} f_{state,\sigma} \qquad \sigma = 1, ..., \Sigma \tag{4.196}$$

$$f_{State,\sigma} \leq F_{state,\sigma} \cdot \vec{v}_{state,\sigma} \qquad \sigma = 1, ..., \Sigma \tag{4.197}$$

$$f_{State,\sigma} \leq \Theta_{state,\sigma} \cdot M \qquad \sigma = 1, ..., \Sigma \tag{4.198}$$

mit einer ausreichend großen Konstante M und der Indikatorvariable $\Theta_{state,\sigma}$ für den jeweiligen Zustand σ. Eine Minimierung kann dargestellte werden als

$$min \sum_{\sigma=1}^{\Sigma} f_{state,\sigma} \qquad \sigma = 1, ..., \Sigma \tag{4.199}$$

$$f_{State,\sigma} \geq F_{state,\sigma} \cdot \vec{v}_{state,\sigma} \qquad \sigma = 1, ..., \Sigma \tag{4.200}$$

$$f_{State,\sigma} \leq \Theta_{state,\sigma} \cdot M \qquad \sigma = 1, ..., \Sigma \tag{4.201}$$

wobei die Formulierung für eine Minimierung nur gilt, wenn die Ersatzvariablen $f_{state,\sigma}$ aller Zielfunktionen ausschließlich nichtnegative Werte annehmen können.

KAPITEL 5

Packingentwurfssprachen in Anwendung

In den vorangegangenen Kapiteln wurden die wichtigsten Anforderungen an ein universelles und automatisiertes Packingframework erarbeitet. Diese sind neben einer konsistenten Wissensrepräsentation die Domänenunabhängigkeit, die Flexibilität und einfache Erweiterbarkeit, die Modularität sowie die komfortable Integrationsfähigkeit in den Gesamtentwurf und nicht zuletzt die Visualisierungsfähigkeit der Ergebnisse. Die im vorliegenden Kapitel vorgestellten Anwendungsfälle haben zum Ziel, die Umsetzung dieser Forderungen in dem entwickelten Packingframework zu demonstrieren. Die konsistente Wissensrepräsentation ergibt sich dabei aus der Verarbeitung von Rand- und Nebenbedingungen unterschiedlichster naturwissenschaftlicher Gebiete, welche in den jeweiligen Beispielen gesondert erläutert werden. Die Verwendung von unterschiedlichen Optimierungsstrategien und Optimierungssoftwares zur Lösung der Layout- bzw. Packingprobleme sowie deren Kombination in den jeweiligen Beispielen wird erst durch die geforderte und umgesetzte Modularität ermöglicht. Die Forderung nach Erweiterbarkeit intendiert eine einfache Anpassungsfähigkeit der Packingentwurfssprache über das Hinzufügen von problemspezifischen Anforderungen. In einigen der folgenden Beispiele werden neben den im Framework bereits vorhandenen Randbedingungen auch weitere Anforderungen in Form von zusätzlichen Constraints modelliert, worin sich die Erweiterbarkeit offenbart. Die Integrationsfähigkeit und die Visualisierungsfähigkeit dagegen werden auf ganz natürliche Weise in allen der drei Beispiele aufgezeigt, da in jeder der genannten Anwendungen der Packingprozess in einen bestehenden Gesamtentwurf integriert wurde und die Ergebnisse in Form von Geometrien visualisiert sind. Die Unabhängigkeit von einzelnen Domänen soll durch die Verwendung von Beispielen aus unterschiedlichsten Branchen wie dem Luftfahrtingenieurwesen, dem Raumfahrtingenieurwesen und dem Bauwesen bzw. der Architektur erwiesen werden.

Das Kapitel *5.1 Gebäudepacking und -layouting in der Architektur* befasst sich mit dem Layout von Gebäuden im Bereich des Bauwesens. Für eine Einführung wird im ersten Teil des Abschnitts eine akademische Gebäudeplanung mithilfe der Packingentwurfssprache realisiert. Die Lösung des modellierten Packingproblems erfolgt dabei in Form einer globalen Optimierung unter Verwendung einer angeschlossenen Optimierungssoftware. Im vorgestellten Anwendungsbeispiel wird auf die Diskrepanz zwischen formalisierbaren und nicht-formalisierbaren Optimierungskriterien eingegangen und die Vielfältigkeit von mathematischen Formulierungen für physikalische Randbedingungen diskutiert. Im weiteren Verlauf des Abschnitts wird vor dem Hintergrund der Flexibilität die Eingliederung des automatisierten Packings in eine Variantengenerierung vorgestellt. Hier liegt der Fokus auf der Fähigkeit der Packingentwurfssprache den Umfang der Optimierung innerhalb des Packingprozesses problemabhängig anzupassen

und damit ein Wechselspiel zwischen einer Optimierung und einer Entwurfsraumexploration zu unterstützen. Im letzten Teil des vorgestellten Beispiels soll dem Leser anhand einer nachfolgenden Grundrissplanung innerhalb eines Gebäudes verdeutlicht werden, inwieweit die Modellierung von Packingprozessen durch das entwickelte Framework vereinfacht wird. Hierzu wird explizit und detailliert auf die vom Framework automatisiert erstellten Optimierungsmodelle eingegangen sowie notwendige manuelle Erweiterungen genauer erörtert.

In *5.2 Packing in einem Flugzeugtragflügel* wird am Beispiel eines Tragflügels vor allem die mühelose Integration des Packings in den Gesamtentwurf demonstriert. Der Gesamtentwurf beinhaltet dabei die Verteilung von Flugzeugkomponenten im Tragflügel und deren anschließende Verrohrung und Verkabelung. Im ersten Teil des Anwendungsbeispiels wird eine Optimierungsstrategie für den automatisierten Gesamtsystementwurf mit eingebettetem Packingverfahren vorgestellt, welche mithilfe einer Optimierungssoftware iterativ alle am Gesamtentwurf beteiligten Entwurfssprachen aufruft. Die vorgestellten Ergebnisse stammen aus einem Forschungsprojekt, dessen Ergebnisse bereits veröffentlicht wurden [Dinkelacker et al., 2021]. Der zweite Teil des Anwendungsbeispiels befasst sich mit der isolierten Konfiguration der elektronischen Komponenten innerhalb des Tragflügels. Für die Lösung dieser Packingaufgabe kommt die Partikelschwarmoptimierung zum Einsatz, welche von dem während der vorliegenden Arbeit entwickelten und implementierten Optimierungsframework bereitgestellt wird.

Die Einteilung des Gesamtentwurfs in einzelne Teilmodelle und die Reihenfolge, in der sie gelöst werden, ist in der Regel nicht eindeutig. Während die Bauteilpositionierung und die Verrohrung in Abschnitt 5.2 nacheinander ausgeführt werden, befasst sich *5.3 Packing und Piping auf einem Satellitenantriebspaneel* als drittes und letztes Anwendungsbeispiel der vorliegenden Arbeit im Kontext der Auslegung eines Satellitenantriebspaneels mit der Überführung der Verrohrung in ein Packingproblem, das parallel zur Bauteilpositionierung gelöst werden soll. Um dies umzusetzen, werden sowohl die Satellitenkomponenten als auch deren Verrohrung zunächst auf ein zweidimensionales Problem reduziert, welches in eine mathematische Formulierung überführt wird. Die nachfolgende Positionierung auf einem zu minimierenden Satellitenantriebspaneel wird in Form eines Optimierungsproblems an eine externe Optimierungssoftware übergeben und mithilfe einer globalen Optimierung berechnet. Die Möglichkeit der zusätzlichen Modellierung der Verrohrung als Packingproblem verdeutlicht hierbei den generischen Charakter der Packingentwurfssprache. Durch die Einbettung des Packings in eine vorhandene Satellitenantriebspaneel-Entwurfssprache wird zudem auch in diesem Beispiel die komfortable Integrationsfähigkeit in den Gesamtentwurf aufgezeigt.

An dieser Stelle sei der Leser außerdem darauf hingewiesen, dass alle im Folgenden ausgeführten Berechnungen auf einem Dell Precision 7550 mit einem Intel®Core™i7-10875H CPU mit 2.3 GHz Arbeitstakt, 128 GB Arbeitsspeicher und einer NVIDIA®Quadro T2000 Grafikkarte ausgeführt werden. Bei Verwendung einer mathematischen globalen Optimierung kommt die Software Gurobi™Optimizer [Gurobi Optimization, LLC, 2022a] zum Einsatz. Zur Optimierung durch Aufruf der Entwurfssprache von außen wird die Prozessintegrations- und Entwurfsoptimierungs-Plattform Optimus®[Noesis Solutions, 2022] eingesetzt.

5.1 Gebäudepacking und -layouting in der Architektur

Der erste Anwendungsfall für das entwickelte und implementierte Framework zur Lösung von Packing- und Layoutproblemen spielt sich im Bereich der Architektur ab. Architektonische Fragestellungen sind deshalb von großem Interesse, weil sie mathematisch abbildbare und damit formalisierbare Nebenbedingungen mit nicht formalisierbaren Forderungen beispielsweise

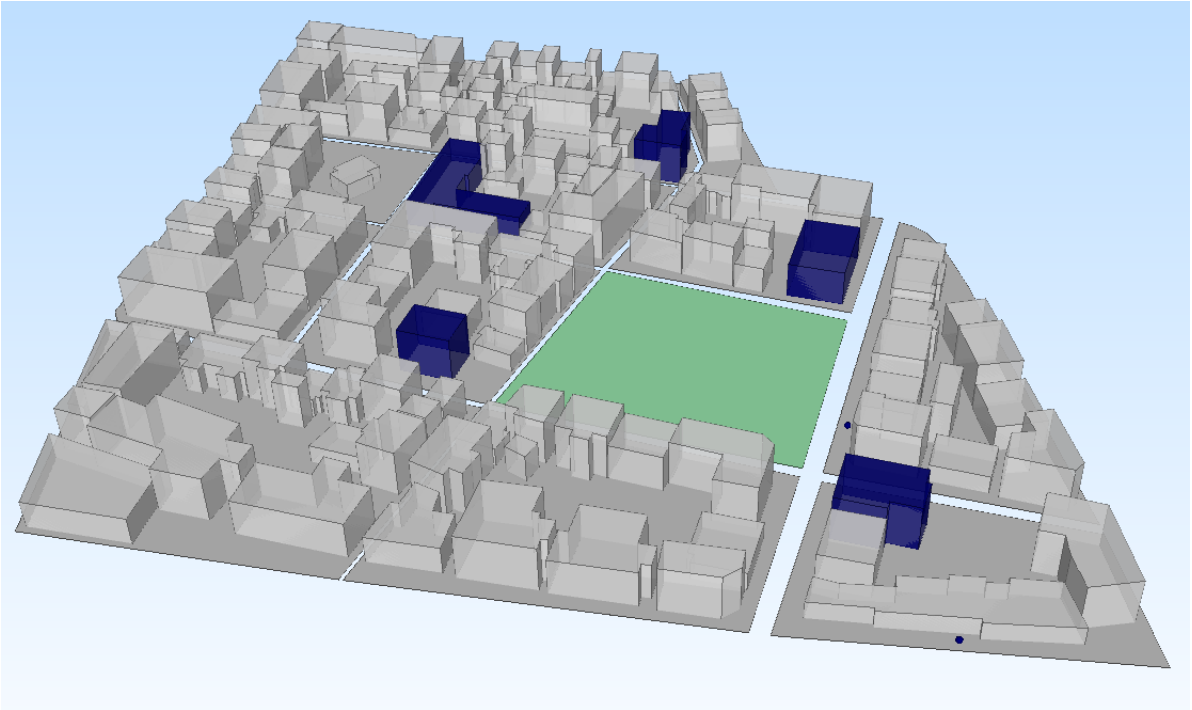


Abbildung 5.1: Bauplatz innerhalb des Stadtviertels inklusive der infrastrukturell wichtigen Einrichtungen und Verkehrspunkte.

aus dem Bereich der Ästhetik kombinieren. Im vorliegenden Beispiel soll aufgezeigt werden, wie ein Wechselspiel zwischen automatisierter Optimierung und manueller Auswahl digital unterstützt werden kann. Die akademische Problemstellung umfasst die Auslegung eines Systems von Gebäudekomplexen für ein frei gewordenes Grundstück innerhalb eines Stadtviertels am Standort Süddeutschland. Als Auslegungszeitpunkt soll der 01. Mai gewählt werden. Ein weiteres Beispiel knüpft direkt an das erste an und befasst sich mit der Auslegung des Gebäudeinneren, indem die Grundrissgestaltung als Layoutproblem formuliert wird. Das Beispiel soll vor allem verdeutlichen, in welchem Ausmaß das entwickelte Framework den/die Benutzer/-in unterstützen kann und wie einfach es möglich ist, erweiternde Aspekte zu berücksichtigen.

Der Startpunkt für das erste Beispiel ist ein Modell des zu bebauenden Grundstücks innerhalb des Stadtviertels inklusive aller infrastrukturell bedeutender Einrichtungen und Verkehrspunkte. Das Modell kann Abb. 5.1 entnommen werden. Die infrastrukturellen Einrichtungen können beispielsweise Einkaufsmöglichkeiten sowie öffentliche Gebäude für Kultur und Sport umfassen und sind im dargestellten Modell dunkelblau eingefärbt. Ebenfalls dunkelblau eingefärbt und als Punkte dargestellt sind eine Bushaltestelle und die Haltestelle der Stadtbahn. Die Anforderungen geben neben der Geometrie des Baulands insgesamt fünf Gebäude vor, deren gewünschte Flächen im einzelnen bereits feststehen sowie deren partielle Adjazenzen untereinander. Weiterhin müssen gesetzliche Randbedingungen beachtet werden, welche die Mindestabstände der Gebäudewände zueinander spezifizieren. Außerdem wird der Stil der Gebäudegrundrisse vorgegeben. Hierbei kann ein einzelnes Gebäude aus einem Komplex mit bis zu drei zueinander versetzten Blöcken bestehen. Die architektonischen Vorgaben geben vier ästhetische Grundformen vor, welche als I-, L-, Γ- oder V-Form bezeichnet werden. Hierzu gibt Abb. 5.2 einen Überblick. Die Versetzungstiefe sowie die Tiefe eines einzelnen Blocks wird für die Auslegung vorgegeben. Variabel sind die Maße entlang der Gebäudelänge und die Anzahl der Stockwerke.

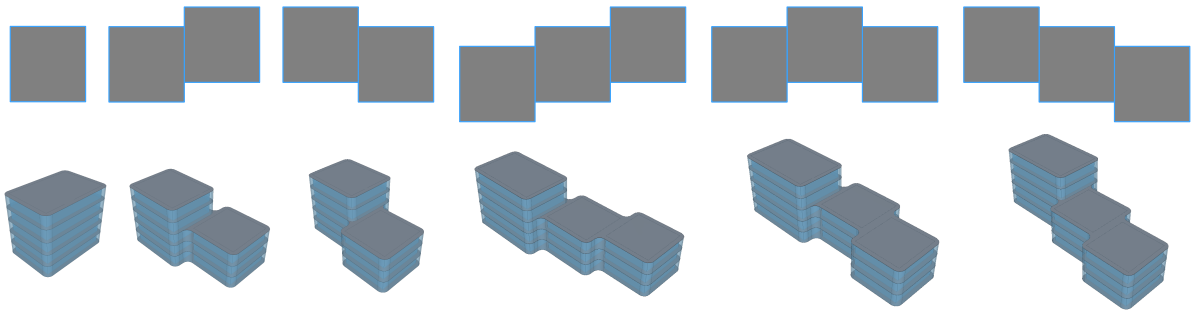


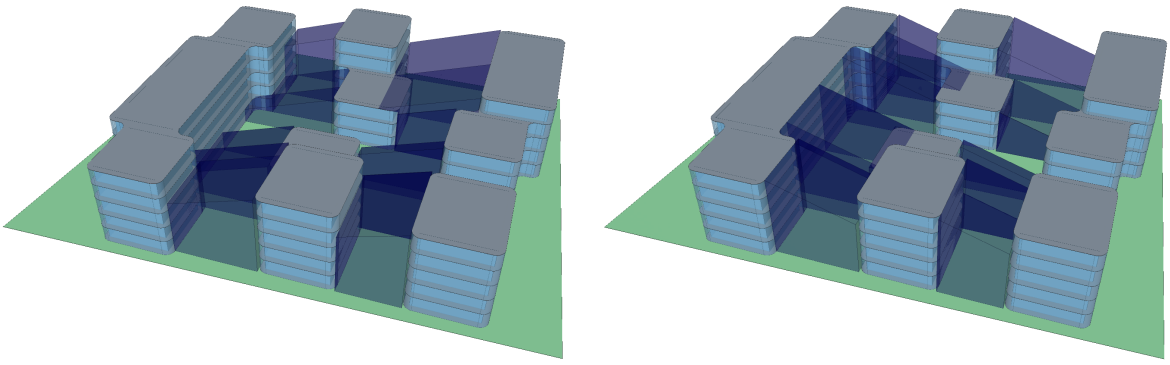
Abbildung 5.2: Ästhetische Gebäudevarianten. V.l.n.r.: 1-Block I-Form, 2-Block L-Form, 2-Block Γ-Form, 3-Block L-Form, 3-Block V-Form, 3-Block Γ-Form.

Als Bewertungskriterien werden eine möglichst geringe durchschnittliche Entfernung zu den definierten Einrichtungen innerhalb des Stadtviertels genannt, sowie eine möglichst geringe Entfernung der adjazenten Gebäude untereinander und schließlich eine möglichst große Sichtweite aus den Gebäuden heraus. Zunächst werden die genannten Parameter formalisiert. Der durchschnittliche Abstand zu allen infrastrukturellen Einrichtungen ergibt sich aus den Abständen der Mittelpunkte der Gebäudeblöcke zu den Eingangskoordinaten der jeweiligen Einrichtung. Aufgrund der näherungsweisen Orthogonalität des Straßennetzes wird die Manhattan-Distanz für die Abstandsberechnung verwendet. Die Abstände zwischen adjazenten Gebäuden ergeben sich aus der euklidischen Distanz zwischen den Mittelpunkten der jeweiligen Gebäude. Für die Sichtweite wird für jedes Gebäude dessen kritischer, oder in anderen Worten minimal auftretender, Sichtabstand normal zur Außenwand gewertet. Für das konkrete Modell bedeutet dies den Einsatz der Manhattan-Distanz-Funktion und der euklidischen Distanz-Funktion. Für die Berechnung der Sichtweite wird der Mindestabstandsparameter aus der Kollisionsvermeidung verwendet, welcher nicht vorgegeben wird, sondern als Variable eingeht. Daneben wird die Min-Funktion zum Auffinden der minimalen Sichtweite verwendet.

Nach der Spezifikation der Berechnungsformeln werden die einzelnen dimensionsbehafteten Zielparameter in dimensionslose Zielkriterien überführt. Der Manhattan-Abstand wird mithilfe der Grundstücksausmaße normiert und wird im Folgenden mit *Infrastrukturelle Norm* κ_I bezeichnet. Der euklidische Abstand zwischen adjazenten Gebäuden wird mithilfe der Grundstücksdiagonalen normiert und wird im Folgenden mit *Norm adjazenter Gebäude* κ_A bezeichnet. Die Sichtweite wird mit dem gesetzlich vorgegebenen Mindestabstand von Gebäudewänden normiert und wird im Folgenden mit *Sichtweitennorm* κ_S bezeichnet. Daraus kann die folgende gewichtete summierte Bewertungsfunktion abgeleitet werden:

$$f = w_I \cdot \kappa_I + w_A \cdot \kappa_A + w_S \cdot \kappa_S \tag{5.1}$$

Neben den zu optimierenden Kriterien werden zusätzlich die Lichtverhältnisse untersucht. Im besten Fall werden diese mithilfe einer Lichtsimulation berechnet, welche innerhalb der Optimierung eingebunden wird. Für eine erste Abschätzung wird die Sonnenstrahlung und der daraus resultierende Schattenwurf lediglich abgeschätzt. Dafür werden drei Tageszeitpunkte definiert, welche den Sonnenstandswinkel und die Sonnenstandshöhe bestimmen. Die Zeitpunkte sind so gewählt, dass die zum gewünschten Auslegungszeitpunkt (01.Mai) auftretende Strahlung genau aus der Ost-, Süd- und Westrichtung kommt. Die Abschätzung ist so implementiert, dass die beschatteten Flächen der Gebäudefensterflächen durch Projektion berechnet werden, wie in Abb. 5.3 dargestellt ist. Die beschatteten sowie die beschattungsfreien Flächen werden mithilfe der gesamten Fensterfläche aller Gebäude normiert.



(a) Oststrahlung morgens

(b) Weststrahlung abends

Abbildung 5.3: Berechnung der beschatteten Fensterflächen mithilfe des Sonnenstandswinkels und der Sonnenstandshöhe an einem 01.Mai am Standort Süddeutschland.

5.1.1 Gebäudepacking zwischen Optimierung und Exploration

Wie bereits angeklungen ist, stehen im vorliegenden Beispiel die formalisierbaren Bewertungskriterien in Kombination mit den nicht formalisierbaren Bewertungskriterien für einen Systementwurf im Fokus. Um beide Arten der Kriterien innerhalb einer Auswertung zu vereinen, wird ein Ansatz vorgestellt, welcher formalisierbare Kriterien optimiert und nicht formalisierbare Kriterien exploriert. Dieser gemischte Ansatz soll den Entwurfsraum so weit einschränken, wie es die formalisierbare Bewertbarkeit des Modells zulässt, gleichzeitig aber auch eine Entwurfsraumexploration durchführen, welche dem Experten oder der Expertin zur subjektiven Auswertung zur Verfügung gestellt werden kann. Die zur Umsetzung entwickelte Entwurfssprache soll im Folgenden vorgestellt werden. Sie umfasst die Problemformulierung, wie sie bereits vorgestellt wurde und deren Optimierung mit der gegebenen Zielfunktion. Gleichzeitig beinhaltet sie eine Variation über die verschiedenen ästhetischen Gebäudevarianten für eine spätere Analyse der hinsichtlich der formalisierten Zielkriterien optimalen Varianten durch einen Architekten oder eine Architektin. Die Parameter zur Variation der ästhetischen Gebäudevarianten sind dabei die Anzahl der Gebäudeblöcke sowie die Gebäudeform. Für die Optimierung wird die in Abschnitt 4.3 vorgestellte Entwurfssprache verwendet, während die Entwurfsraumexploration mithilfe des im *Design Cockpit 43*®integrierten *Variation*-Plugin durchgeführt wird. Darüber hinaus soll am Beispiel einer Lichtberechnung demonstriert werden, wie eine mögliche simulationsbasierte Auswertung innerhalb des Prozesses integriert werden kann. Diese soll weder in die Optimierung eingehen, noch in die Entwurfsraumexploration und soll stattdessen für jede Variante ausgerechnet werden, um die manuelle Entscheidungsfindung zu unterstützen. Da echte Simulationen rechen- und zeitaufwändig sind, wird diese aufgrund der zur Verfügung stehenden Hardware durch eine einfache Abschätzung ersetzt, welche für die Demonstration der Integrationsmöglichkeit ausreicht.

Das Klassendiagramm der Entwurfssprache ist in Abbildung 5.4 dargestellt. Im Zentrum steht die Optimierung für jede ästhetische Variante, welche durch die Klasse `Optimization` repräsentiert wird. Über die Klasse `BuildingConfiguration` kann die Problemstellung angepasst werden, denn hier werden alle bereits beschriebenen Randbedingungen spezifiziert. Dies betrifft die Vorgabe einer Gebäudeanzahl, die Auswahl der mit der Gebäudefläche korrelierenden Freiheitsgrade, die Definition der einzelnen Zielfunktionsgewichtungen sowie die Eingabe des minimalen vorgeschriebenen Gebäudeabstandes und des ästhetischen Verschiebungsparameters. Die Klasse bildet den Hauptinput für die Optimierung. Gebäudespezifische Eingangspa-

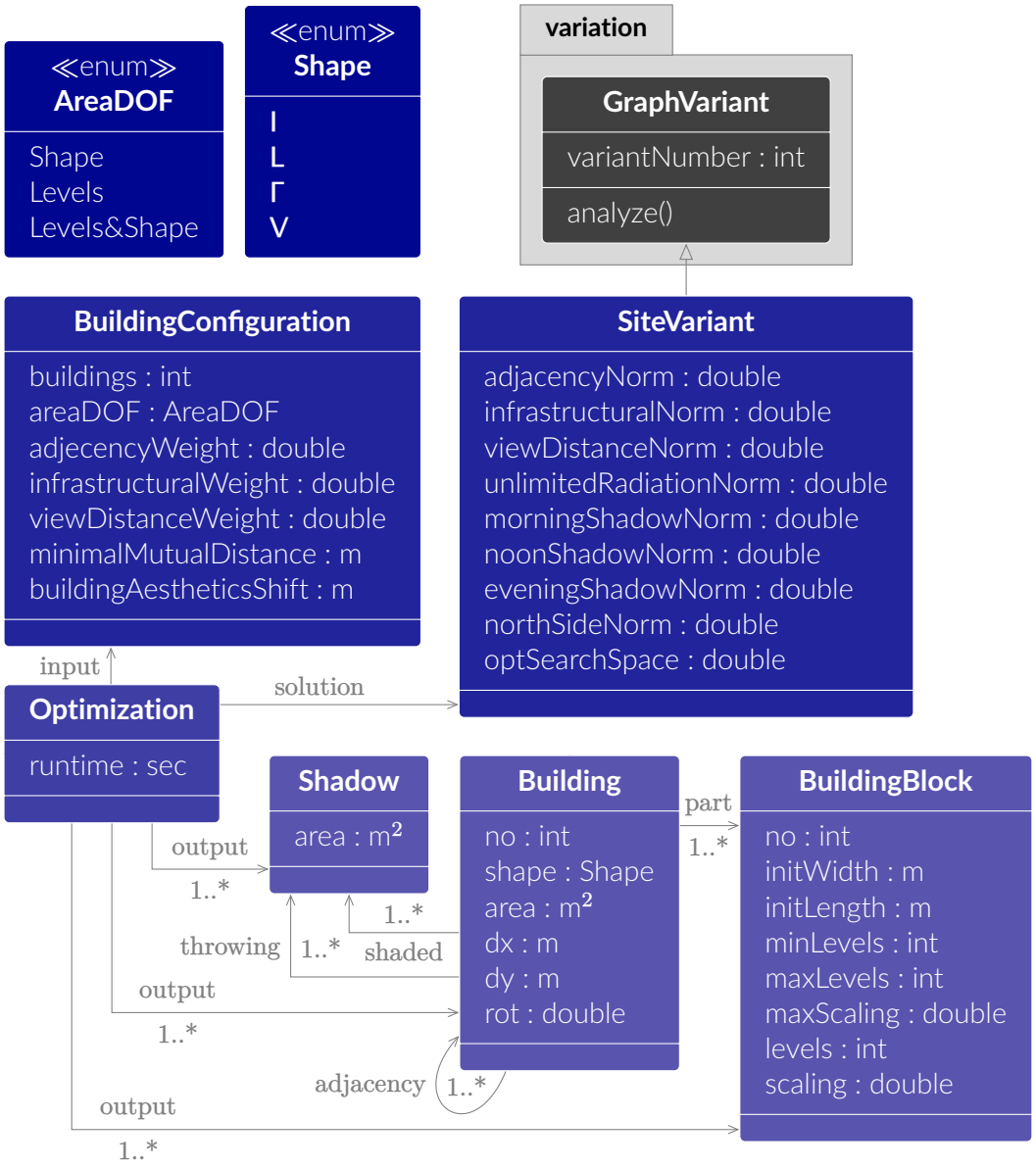


Abbildung 5.4: Klassendiagramm der Entwurfssprache für die Gebäudeplanung.

rameter werden über die Klassen `Building` und `BuildingBlock` definiert. Hier werden die gewünschten Gebäudeflächen spezifiziert sowie die initiale Gebäudelänge und -tiefe vorgegeben wie auch die minimalen und maximalen Werte für die Skalierung eines Gebäudeblocks und die Auslegung der Stockwerke vorgegeben. Über die Assoziation `adjacency` können die adjazenten Gebäudepaare spezifiziert werden. Die Klasse `Shadow` wird für die Abschätzung der Sonneneinstrahlung benötigt. Die Klasse `SiteVariant` repräsentiert eine optimierte Variante und beinhaltet alle optimierten und noch manuell zu vergleichenden Kriterien. Für den Einsatz des *Variation*-Plugins muss diese von der aus dem Plugin zur Verfügung gestellten Klasse `GraphVariant` erben.

Das zugehörige Aktivitätsdiagramm ist in Abbildung 5.5 zusammengefasst. Es umfasst alle Spezifikationsschritte sowie die Entwurfsraumexploration auf Basis der Variationstreiber und die Optimierung aller daraus entstehenden Gebäudevarianten. Vor jedem Explorationsschritt werden zunächst die validen Explorationsvarianten auf Basis der vorgegebenen Randbedin-

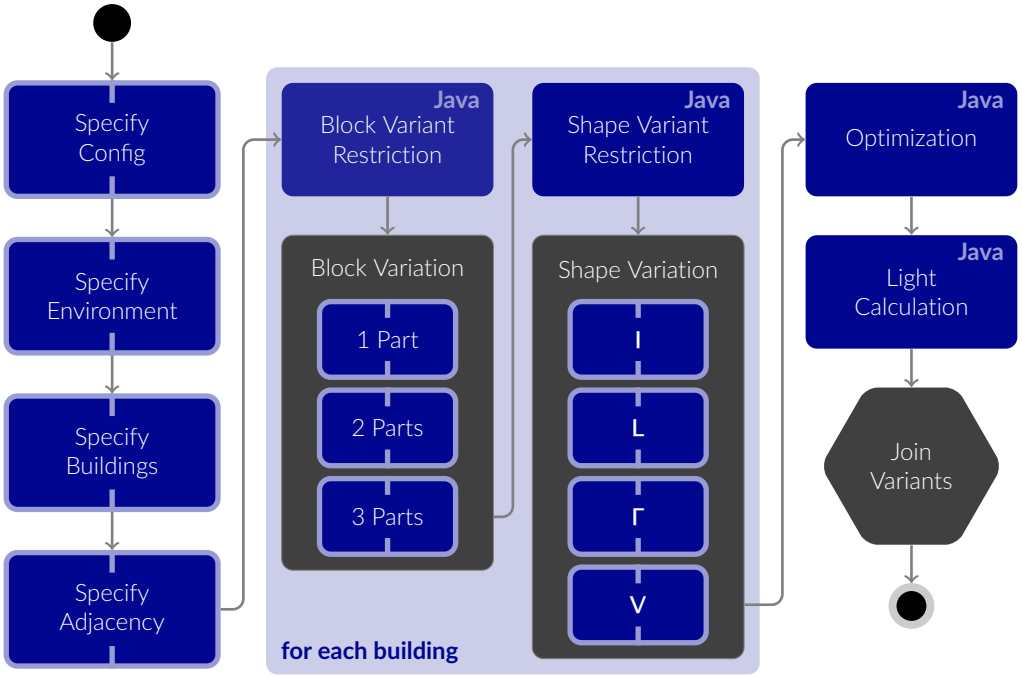


Abbildung 5.5: Aktivitätsdiagramm der Entwurfssprache für die Gebäudeplanung.

gungen eingeschränkt. So können schon vor der Exploration Parameterzusammensetzungen ausgeschlossen werden, welche möglicherweise trotz Invalidität eine zeitaufwändige Optimierung auslösen würden. Zwar bricht diese bei Identifikation der Unlösbarkeit sofort ab, der Aufbau des Optimierungsproblems und der Aufruf der Optimierungssoftware nehmen dennoch Rechenzeit in Anspruch, welche so eingespart werden kann. Ein Beispiel wäre eine gewünschte Gebäudefläche, welche mit den vorgegebenen Grenzwerten für die Skalierung und die Anzahl der Stockwerke nicht erreicht werden kann. Diese invalide Variante kann einfach überprüft und vor der Exploration abgefangen werden. Gleiches gilt für die Variation der Gebäudeformen, da nicht jede Blockanzahl-Gebäudeform-Kombination zusammenpasst. In dem vorgestellten Anwendungsfall mit fünf Gebäuden mit jeweils bis zu drei Blöcken und den vier genannten Gebäudeformen sind theoretisch $(4 \cdot 3)^5$, also ca. 250000 Varianten für die Gesamtanlage möglich, unter Beachtung der Restriktionen durch die validen Fläche-Blockanzahl-Kombinationen und die validen Blockanzahl-Gebäudeform-Kombinationen schrumpfen diese auf 225 Varianten für die nachfolgende Optimierung zusammen.

Im Folgenden sollen die Ergebnisse des ausgewählten Szenarios vorgestellt werden. Für die Ausführung wurde jeder Optimierungslauf auf 15 Minuten begrenzt. Durch die Einschränkung der Rechenzeit werden keine vollständigen globalen Optimierungen durchgeführt. In diesem Kontext ist die sog. *MIP Gap* relevant, welche angibt, wie groß der auf den aktuellen Zielwert bezogene prozentuale Abstand zwischen der globalen oberen und unteren Zielschranke ist. Die *MIP Gap* kann im weitesten Sinne als Maß für den Anteil des bereits untersuchten Entwurfsraums und damit für die Zuverlässigkeit der gefundenen Lösung hinsichtlich eines globalen Optimums verwendet werden. Der so definierte Anteil des bei der Optimierung untersuchten Entwurfsraums wird in der Auswertung farblich gekennzeichnet. Darüber hinaus werden für die Auswertung alle normierten Kriterien mit dem Faktor 100 multipliziert, sodass sich prozentuale Angaben ergeben. Zunächst sollen die Optimierungen aller Varianten betrachtet werden. Die Abbildungen 5.6, 5.7 und 5.8 geben eine Übersicht über die Ergebnisse für die einzelnen Kriterien, welche über den optimierten Varianten der Gesamtanlage aufgetragen sind. Es wird

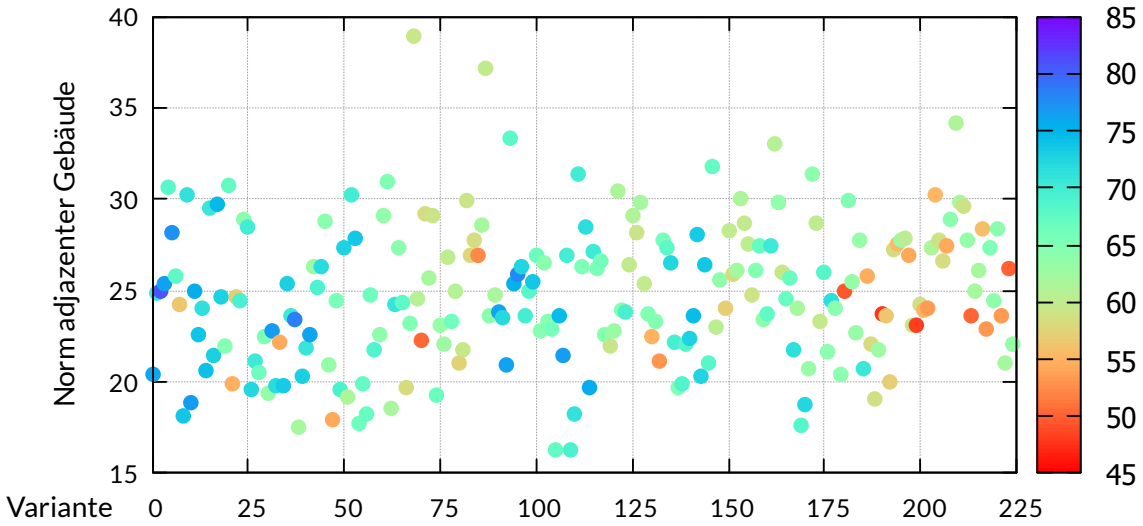


Abbildung 5.6: Mittlere normierte Abstände adjazenter Gebäude.

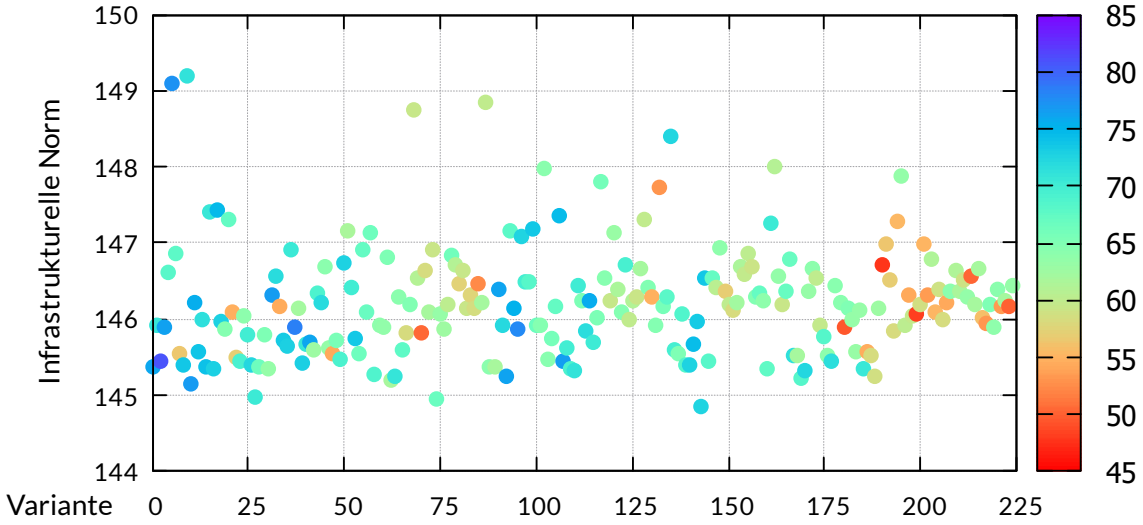


Abbildung 5.7: Mittlere normierte Abstände zu öffentlichen Einrichtungen.

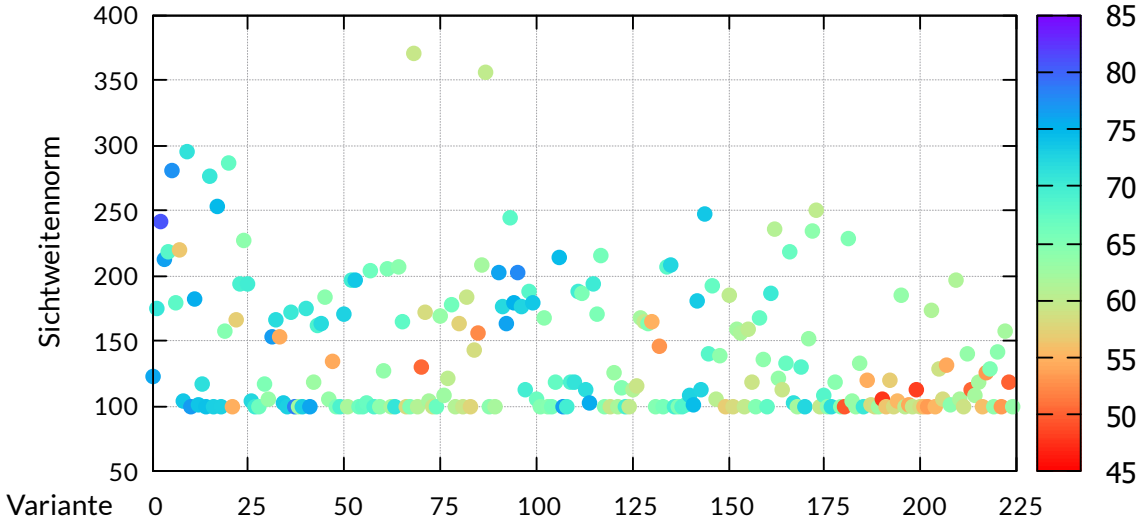


Abbildung 5.8: Mittlere normierte kritische Sichtweite.

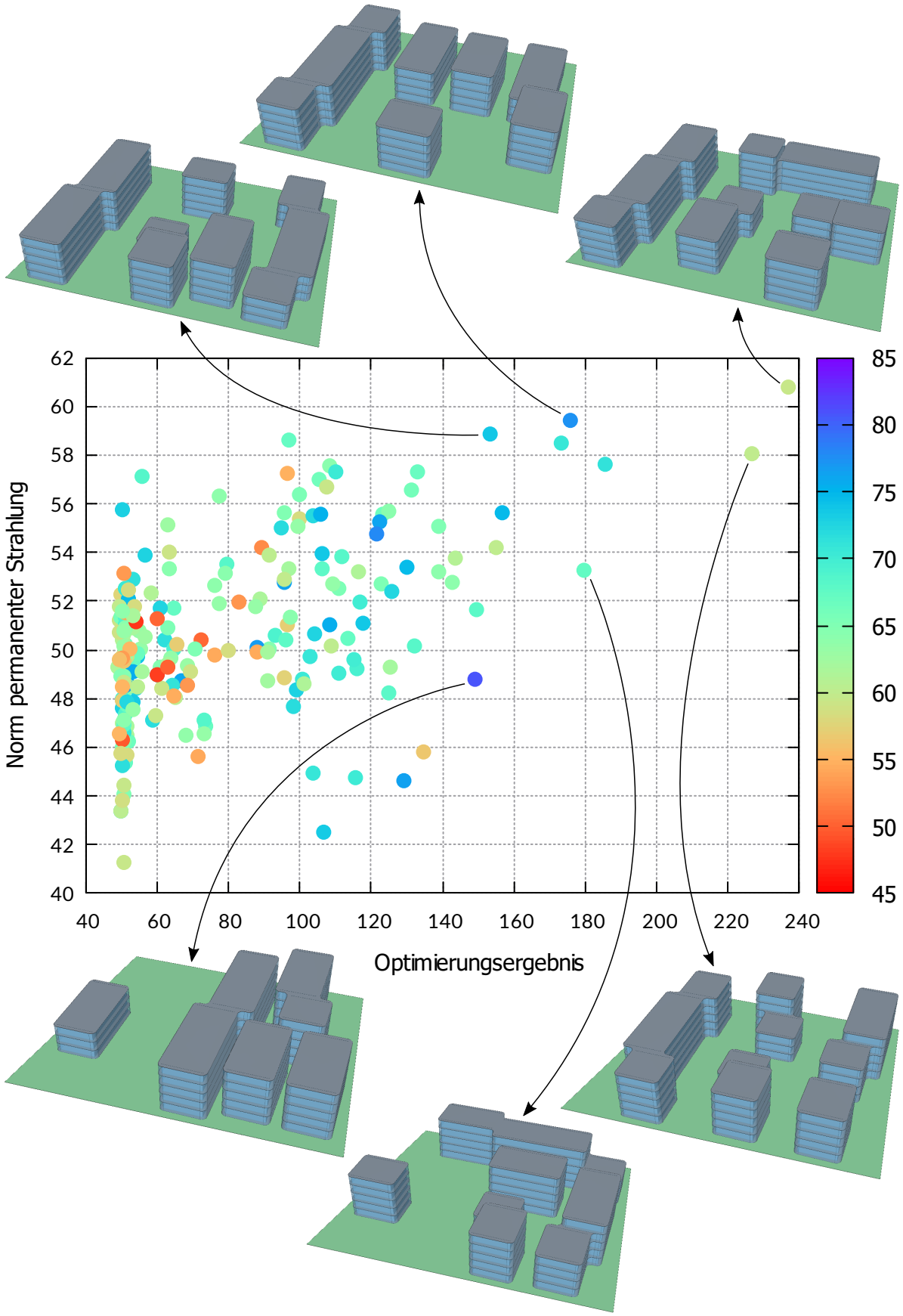


Abbildung 5.9: Permanenter Sonnenstrahlung ausgesetzte normierte Fensterflächen.

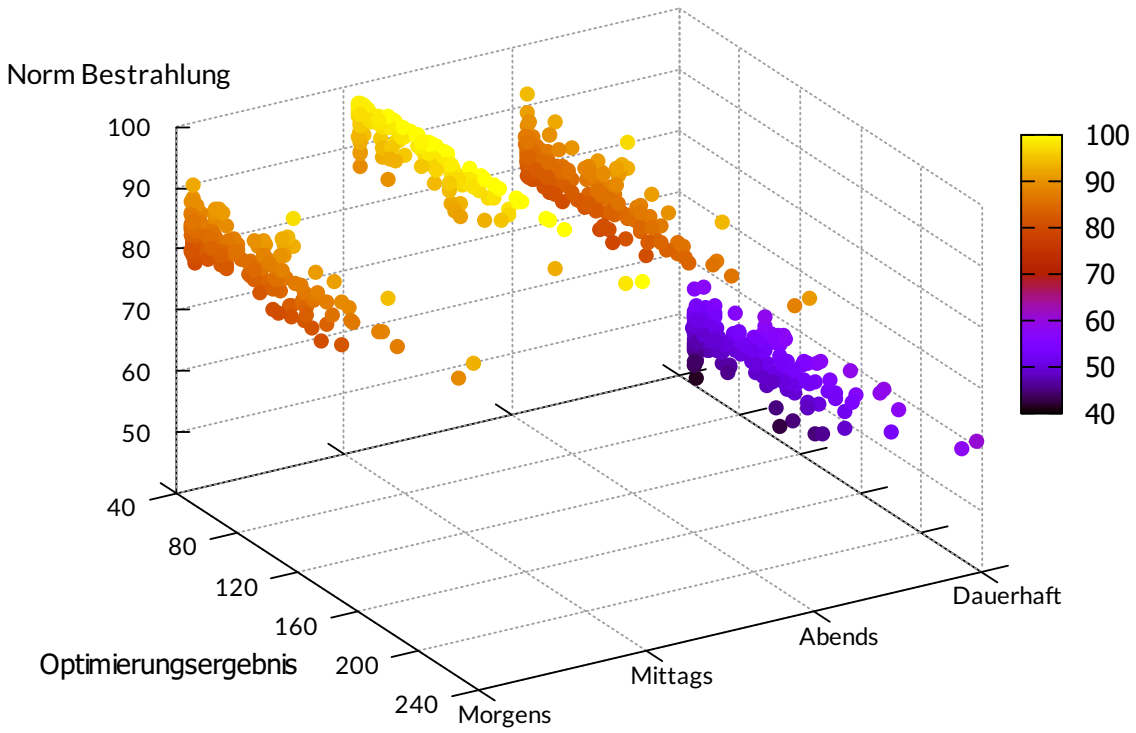


Abbildung 5.10: Prozentuale Angabe der besonnten normierte Fensterflächen.

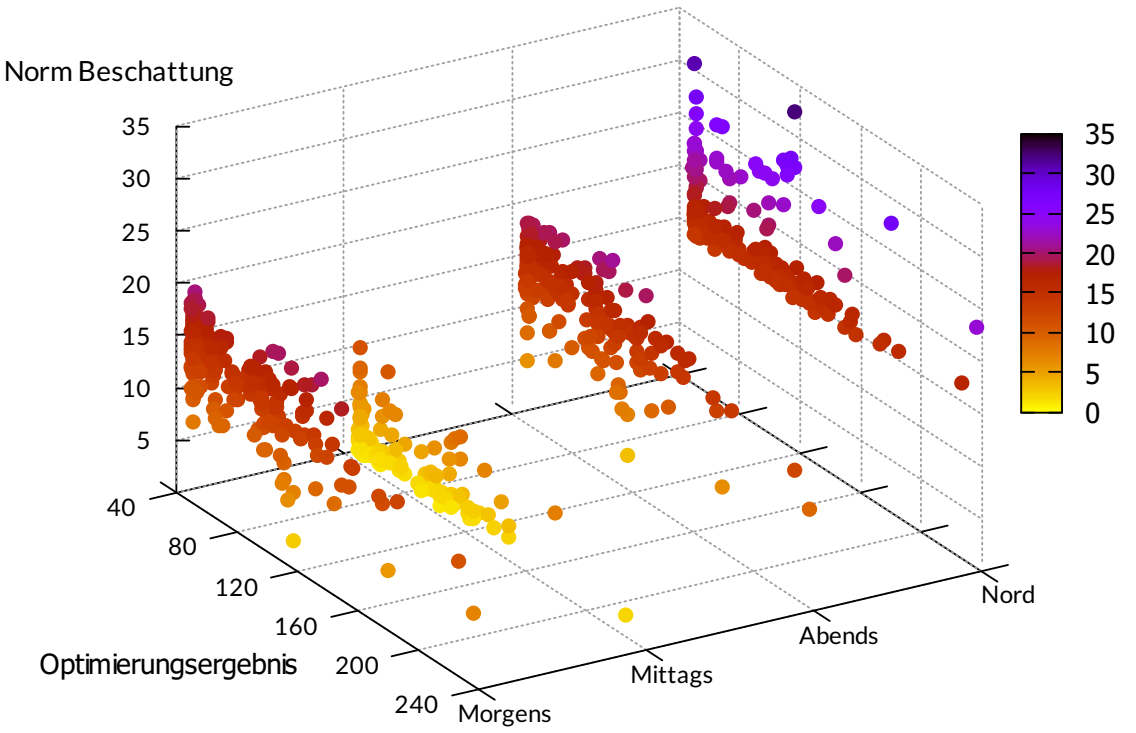


Abbildung 5.11: Prozentuale Angabe der beschatteten normierten Fensterflächen.

deutlich, dass die Optimierungsprobleme unterschiedliche Komplexitäten aufweisen, was sich aus den schwankenden Werten für den untersuchten Anteil des Entwurfsraumes ableiten lässt. Die Interpretation der Ergebnisse muss diese Unsicherheit berücksichtigen.

Nachdem die formalisierten Kriterien optimiert wurden, soll die abgeschätzte Sonneneinstrahlung näher betrachtet werden. Da angenommen wird, dass die Bewertungsfunktion für die formalisierten Kriterien gilt, müssen diese nicht mehr separat betrachtet werden, sodass stattdessen das Ergebnis der jeweiligen Optimierung als Richtwert dient. Die Sonnenstrahlung kann also über den Optimierungsergebnissen aufgetragen werden, wie in Abb. 5.9 dargestellt ist. Die Interpretation der Ergebnisse, d.h. die Bewertung, ob die Sonneneinstrahlung minimiert oder maximiert werden sollte, obliegt nun dem/der Anwender/-in der Auslegungsentwurfssprache. Durch die vorab getätigte Optimierung kann die manuelle Auswertung stark vereinfacht werden, was den Hauptvorteil der Vorgehensweise ausmacht. Durch die einfachere Interpretierbarkeit der messbaren Ergebnisse, kann damit auch die Entscheidungsfindung unter Berücksichtigung der subjektiven ästhetischen Kriterien vereinfacht werden. In Abb. 5.9 sind vielversprechende Ergebnisse visualisiert, welche sowohl für eine gewünschte hohe Sonneneinstrahlung infrage kommen oder für eine gewünschte mittlere Sonneneinstrahlung. Auch hierbei kann für die Interpretation der Ergebnisse die Unsicherheit der Optimierung hinsichtlich der Suboptimalität berücksichtigt werden.

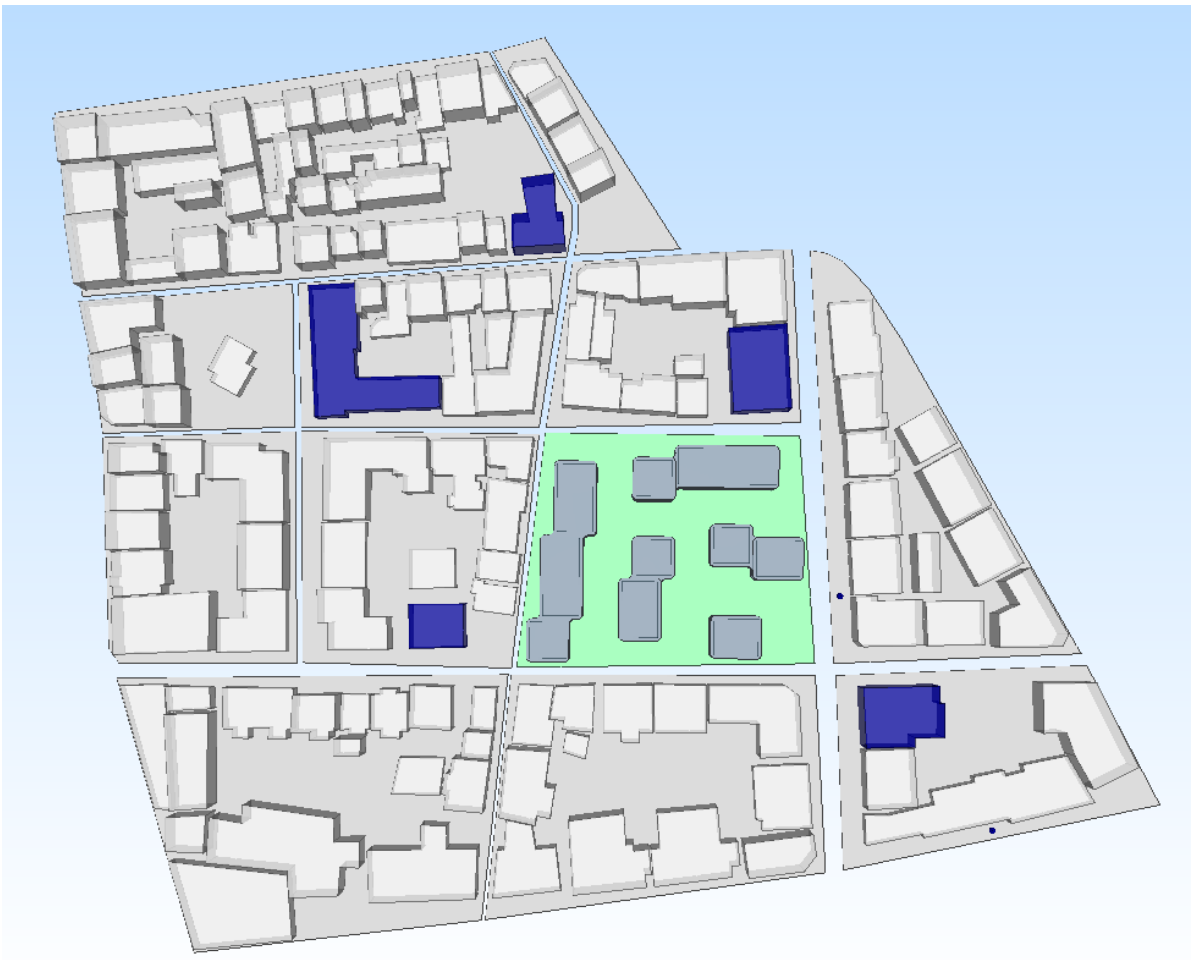


Abbildung 5.12: Ausgewählte Lösung als bester Kompromiss aus formalisierten, berechneten und subjektiven Kriterien.

Die Auswertung der permanent beschienenen Fensterflächen ermöglicht eine vereinfachte Analyse. In vielen Fällen können aber auch andere Parameter der Lichtverhältnisse eine Rolle spielen. Für eine umfassendere Auswertung der Lichtverhältnisse können die normierten bestrahlten Fensterflächen zu den einzelnen Tageszeitpunkten der Berechnung untersucht werden. Diese sind Abbildung 5.10 zu entnehmen. Zum Vergleich wurde die normierte Fläche mit dauerhafter Sonneneinstrahlung aus Abb. 5.9 ebenso in das Schaubild eingetragen. Nicht für jeden Anwendungsfall ist die Auswertung der prozentual beleuchteten Fensterfläche zweckdienlich. Sind die Gebäude beispielsweise großflächig nach Süden ausgerichtet, bedeutet das im vorgestellten Szenario gleichzeitig auch eine ähnlich große nach Norden ausgerichtete Fensterfront. Ist die Minimierung der nach Norden ausgerichteten Fläche gewünscht, kann die Auswertung aus Abbildung 5.11 analysiert werden. Hier wird die beschattete normierte Fensterfläche zu den gewählten Tageszeitpunkten dargestellt, sowie die nach Norden ausgerichtete prozentuale Fensterfläche aller Gebäude. Es ist zu beachten, dass in beiden Schaubildern die Färbung ausschließlich zur Verdeutlichung des z-Koordinatenwertes dient und an dieser Stelle keine Auskunft über den zuvor farblich gekennzeichneten untersuchten Anteil des Entwurfsraumes gibt. Mithilfe der vorgestellten Auswertungen kann eine Lösung ausgesucht werden, die den vielversprechendsten Kompromiss aus formalisierten, berechneten und subjektiven Kriterien darstellt. Diese ist abschließend in Abbildung 5.12 visualisiert.

5.1.2 Packingframework zur Layoutoptimierung im Detail

Nachdem ein Layout für den Standort gefunden wurde, sollen in diesem Abschnitt nun die Grundrisse der Gebäude näher betrachtet werden. Während die mathematischen Problemformulierungen der anderen Beispiele durch die Anwendung des Packingframeworks implizit als gegeben betrachtet werden, soll das vorliegende Beispiel hierbei einer genaueren Untersuchung der Modellbildung dienen. Dafür werden die Problemmodelle, welche in den Ontologien des Packingframeworks kodiert sind und durch Aufruf der jeweiligen Funktionen automatisiert aufgebaut werden, im Detail aufgezeigt. In diesem Zuge soll außerdem die Vereinfachung der Modellierung durch die Verwendung des entwickelten Frameworks demonstriert werden.

Für die Definition der Grundrisse wird eine eigenständige Entwurfssprache erstellt. Da die Grundrisserstellung im Allgemeinen planar ist, kann ein zweidimensionales Ersatzproblem für die Optimierung definiert werden. Dieses wird in der beschriebenen Entwurfssprache mithilfe des Packingframeworks modelliert. Das Klassendiagramm, welches die Ontologie der Grundrisse repräsentiert, ist in Abb. 5.13 dargestellt. Der Workflow ist in zwei Schritte gegliedert. Zunächst werden graphische Regeln definiert, in denen alle Randbedingungen für die Layoutoptimierung festgelegt werden. Diese enthalten die Adjazenzen der Räume, vorgegebene Positionen oder Maße, einen zulässigen Raumverhältnisfaktor sowie die minimalen und maximalen Werte der Freiheitsgrade. Bei Ausführung spezifizieren sie den Eingangsgraphen, aus dem nun ein Optimierungsmodell erstellt werden muss. Der Aufbau des eigentlichen Optimierungsmodells findet darauffolgend in einer Javarule statt. Das Aktivitätsdiagramm als Ganzes ist in Abb. 5.14 dargestellt. Die ersten drei Regeln definieren die Grundprobleme, welche untersucht werden sollen. Diese betreffen die Optimierung für ein freistehendes Gebäude (siehe Abschnitt 5.1.2), für einen Reihenendblock im Gebäudekomplex (siehe Abschnitt 5.1.2) und für einen Reihenmittelblock im Gebäudekomplex (siehe Abschnitt 5.1.2). In den folgenden beiden Regeln werden alle von der Gebäudeart unabhängigen Nebenbedingungen spezifiziert. Darauf folgt abschließend die Javarule zum Aufbau des Optimierungsmodells, deren Flussdiagramm Abb. 5.15 entnommen werden kann.

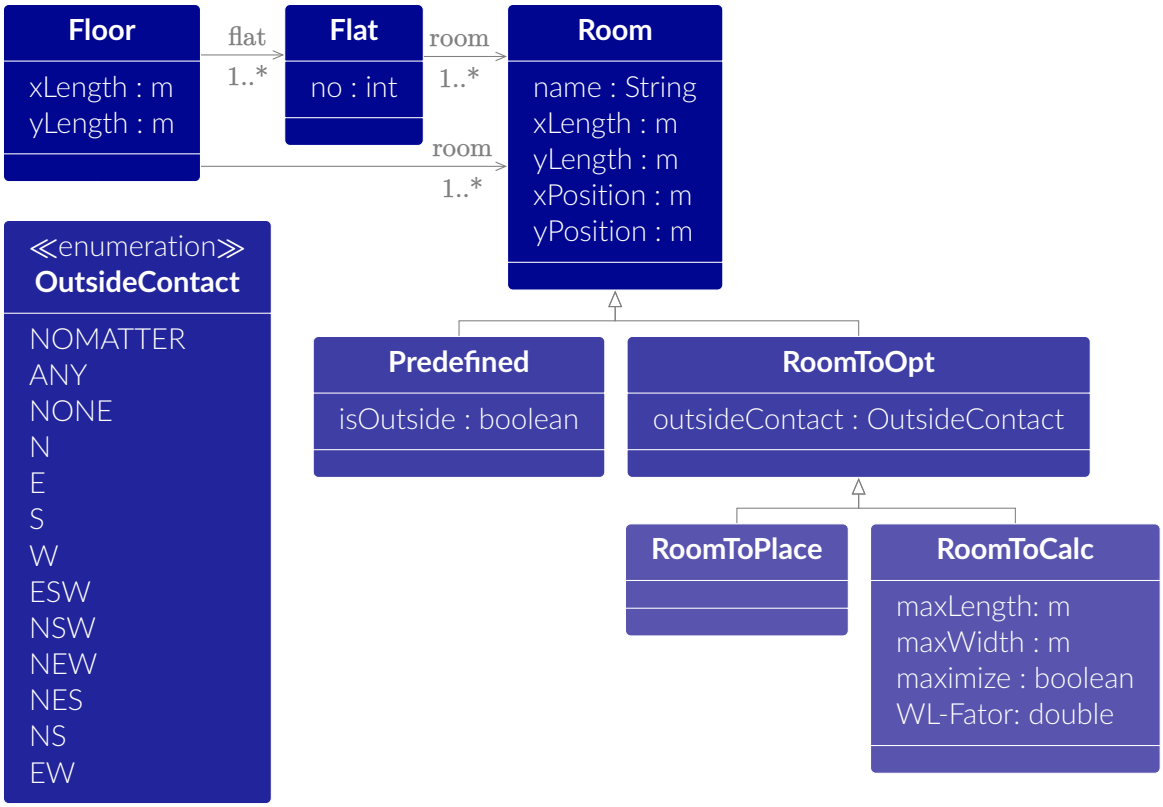


Abbildung 5.13: Klassendiagramm der Entwurfssprache für Layoutoptimierung.

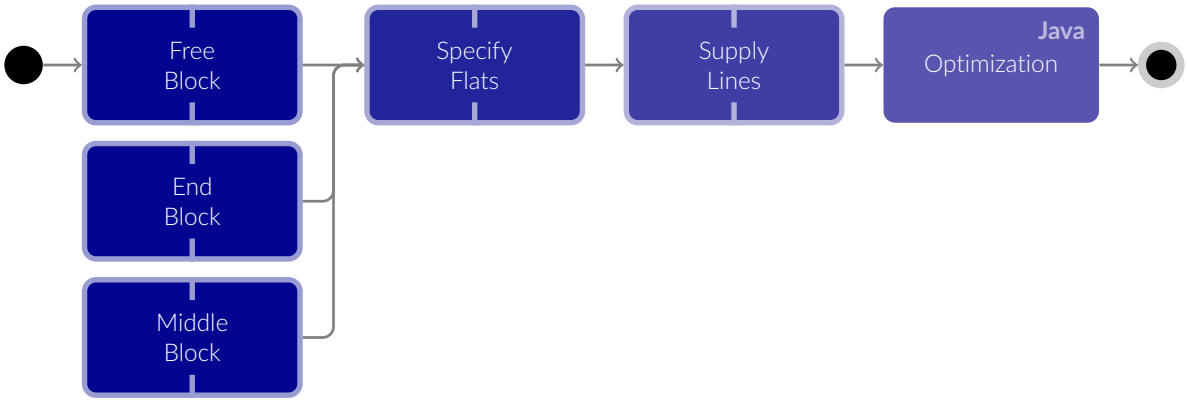


Abbildung 5.14: Aktivitätsdiagramm der Entwurfssprache für Layoutoptimierung.

Die erstellte Entwurfssprache soll im Folgenden auf drei Fragestellungen angewendet werden. Alle der betrachteten Fragestellungen befassen sich mit der Optimierung eines Stockwerkes in einem der im vorigen Abschnitt definierten Gebäude, in dem zwei Wohnungen untergebracht werden sollen. Die Räume der einzelnen Wohnungen werden in der Entwurfssprache so modelliert, dass sie alle gegenseitig über den Flur erreichbar sind. Außerdem werden das Treppenhaus sowie die beiden zu den Wohnungen gehörenden Balkone vorher definiert. Eine weitere Besonderheit ist die Vorgabe, dass die Bäder der beiden Wohnungen nebeneinander liegen müssen, um einen gemeinsamen Versorgungsschacht nutzen zu können. Eine weitere Bedingung ist, dass jeder der Wohnräume, an einer Außenwand gelegen sein muss, sodass eine

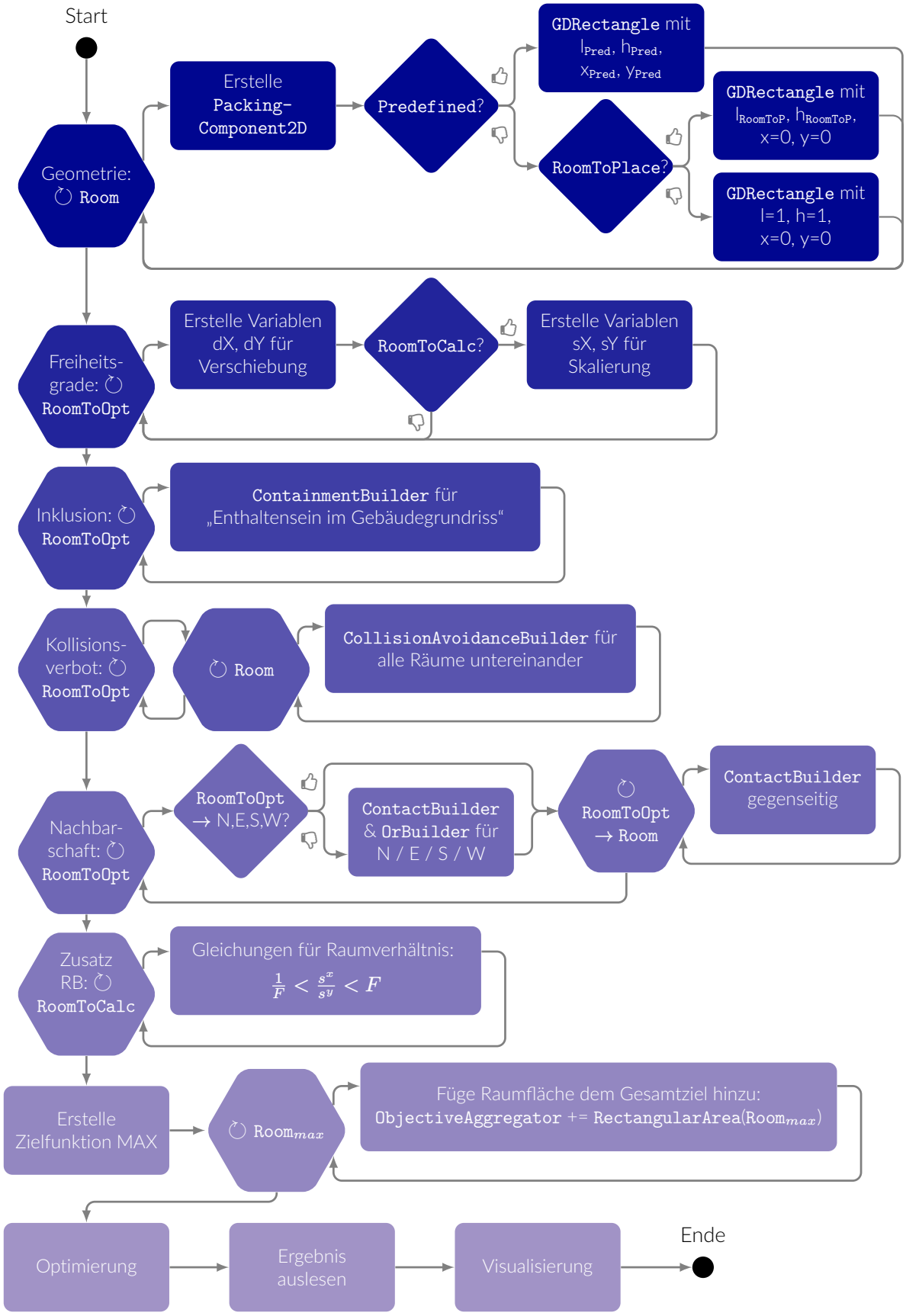


Abbildung 5.15: Workflow bei der Modellierung des Grundriss-Layoutproblems.

Ausleuchtung mit natürlichem Licht sichergestellt werden kann. Ausgenommen sind hierbei ausschließlich die Flure, welche auch durchgängig durch künstliche Lichtquellen versorgt werden können.

Mathematisches Modell für die Grundrissoptimierung

Wie bereits besprochen ist die Erstellung der Grundrisse durch ein zweidimensionales Ersatzproblem repräsentiert, aus welchen sich ein ganzzahlig-linear beschränktes quadratisches Optimierungsproblem ergibt. Für die Darstellung der Gleichungen und Ungleichungen sollen zunächst die folgenden Indizes und Parameter eingeführt werden.

Indizes:

r: Index für den Raum $r = 0, ..., R$
r_{Opt}: Index für den Raum $r_{Opt} \in M_{Opt}$ zur Optimierung
r_{Calc}: Index für den Raum $r_{Calc} \in M_{Calc}$ mit variabler Position und Skalierung
r_{Place}: Index für den Raum $r_{Place} \in M_{Place}$ mit variabler Position
r_{Max}: Index für den Raum $r_{Max} \in M_{Max}$ mit zu maximierender Fläche
r_{Pred}: Index für den vordefinierten Raum $r_{Pred} \in M_{Pred}$
N: Index für die Gebäude-Nordseite
E: Index für die Gebäude-Ostseite
S: Index für die Gebäude-Südseite
W: Index für die Gebäude-Westseite

Parameter:

R: Anzahl aller Räume inklusive der Balkone
M_{Opt}: Menge der Räume zur Optimierung
M_{Calc}: Menge der Räume mit variabler Position und Skalierung
M_{Place}: Menge der Räume mit variabler Position
M_{Max}: Menge der Räume, deren Fläche maximiert werden soll
M_{Pred}: Menge der vordefinierten Räume
l_r, w_r: Länge und Breite des Raumes r
L, W: Länge und Breite des Gebäudes

In der Javarule für den Aufbau des Optimierungsproblems werden zunächst für alle `Room` Instanzen im Entwurfsgraphen Rechtecksgeometrien erstellt. Die Initialmaße der Rechtecke orientieren sich daran, ob sie bereits vorgegeben sind (`Predefined` oder `RoomToPlace`) oder ob ein Raum während der Optimierung dimensioniert werden soll (`RoomToCalc`). Die vorherrschenden Freiheitsgrade werden ebenso durch die Klasse eines Raumes bestimmt, `RoomToPlace` Instanzen müssen lediglich positioniert werden, während `RoomToCalc` Instanzen auch skaliert werden. Für jeden Freiheitsgrad der gegebenen Räume werden nun Optimierungsvariablen eingeführt. Die Initialisierung der Variablen inklusive deren Wertebereichen muss hierbei durch den/die Programmierer/-in der Entwurfssprache kodiert werden. Die Wertebereiche der Variablen können dabei aus dem Entwurfsgraphen ausgelesen werden, da sie in den vorhergehenden Regeln beim Aufbau der Raumstruktur bereits definiert worden sind. Insgesamt ergeben sich die folgenden Freiheitsgrade.

Kontinuierliche Variablen:

x_r, y_r: Position der linken vorderen Ecke des Raumes r
s_r^x, s_r^y: Skalierung des Raumes r in x- und y-Koordinatenrichtung

Einschränkungen der Parameter und Variablen:

$l_r,\, w_r = 1 \qquad \forall\, r \in M_{Calc}$
$x_r,\, y_r = const \quad \forall\, r \in M_{Pred}$
$s_r^x,\, s_r^y = 1 \qquad \forall\, r \in M_{Pred} \vee r \in M_{Place}$

Wie Abb. 5.15 zu entnehmen ist, wird für jeden Raum dessen Enthaltensein innerhalb des Gebäudegrundrisses über die `ContainmentBuilder` Klasse des Packingframeworks sichergestellt. Im Hintergrund werden dabei die folgenden Nebenbedingungen für die Optimierung aufgebaut, welche später an das Optimierungsverfahren übergeben werden. Der/die Anwender/-in muss bei der Programmierung nichts weiter tun, als die automatisch aufgebauten Ungleichungen zu sammeln und an das Gesamtproblem am Ende zu übergeben. Die betreffenden Ungleichungen sind im Folgenden aufgeführt.

Gleichungen für die Inklusion der Räume innerhalb des Gebäudegrundrisses:

$$x_{r_{Opt}} + s^x_{r_{Opt}} \cdot l_{r_{Opt}} \leq x_E \tag{5.2}$$

$$x_W \leq x_{r_{Opt}} \tag{5.3}$$

$$y_{r_{Opt}} + s^y_{r_{Opt}} \cdot w_{r_{Opt}} \leq y_N \tag{5.4}$$

$$y_S \leq y_{r_{Opt}} \tag{5.5}$$

Bei der Positionierung der Räume soll außerdem berücksichtigt werden, dass sich diese nicht überschneiden. Dafür wird für jedes Raumpaar die `CollisionAvoidanceBuilder` Klasse des Packingframeworks aufgerufen. Diese baut automatisiert die notwendigen Ungleichungen inklusive der zusätzlich benötigten Indikatorvariablen auf. Daneben sollen die zuvor in den Regeln der Entwurfssprache definierten Adjazenzen zwischen Räumen berücksichtigt werden, welche aus dem Entwurfsgraphen ausgelesen werden können. Für diese werden durch einen Aufruf der `ContactBuilder` Klasse und Übergabe des Enumerationswertes RectangleContact.CONTACT und der mindestens geforderten Kontaktlänge $length_{overlap}$ weitere Optimierungsmodelle generiert. Der Wert der Kontaktlänge ergibt sich aus der Forderung, dass die Vorgabe einer Adjazenz einen Durchgang zwischen den jeweiligen Räumen zur Folge hat. Die Kontaktlänge orientiert sich daher an den Standard-Türmaßen nach der DIN-Norm inklusive der dazugehörigen Wandöffnungsmaße. Die `ContactBuilder` Klasse kann auch genutzt werden, wenn Nachbarschaften zwischen Räumen und Gebäudeaußenwänden spezifiziert sind. Dann kann einer der Enumerationswerte RectangleContact.BOTTOM (of North), RectangleContact.TOP (of South), RectangleContact.LEFT (of East) oder RectangleContact.RIGHT (of West) als Parameter an die `ContactBuilder` Klasse übergeben werden. Soll nicht eine konkrete Außenwand für den Kontakt vorgegeben werden, sondern eine mögliche Auswahl an Wänden, können die einzelnen Kontakte durch die `OrBuilder` Klasse des Packingframeworks in Zusammenhang gebracht werden. Die `OrBuilder` Klasse ist so konzipiert, dass von allen übergebenen Bedingungen, mindestens eine wahr sein muss (siehe Gl. (5.36)). Die erzeugten mathematischen Systeme sind im Folgenden beschrieben.

Binäre Variablen für die relative Positionierung der Räume:

$$\Theta^{xy}_{r_i,r_j} = \begin{cases} 1 & \text{wenn eine Verschiebung des Raumes } r_i \text{ zum Raum } r_j \text{ in} \\ & \text{Y-Koordinatenrichtung vorliegt,} \\ 0 & \text{wenn eine Verschiebung des Raumes } r_i \text{ zum Raum } r_j \text{ in} \\ & \text{X-Koordinatenrichtung vorliegt} \end{cases} \tag{5.6}$$

$$\Theta^{+/-}_{r_i,r_j} = \begin{cases} 1 & \text{wenn die Verschiebung des Raumes } r_i \text{ zum Raum } r_j \text{ positiv ist,} \\ 0 & \text{wenn die Verschiebung des Raumes } r_i \text{ zum Raum } r_j \text{ negativ ist} \end{cases} \tag{5.7}$$

Gleichungen für das Kollisionsverbot der Räume untereinander:

$$x_r + s^x_r \cdot l_r \leq x_{r_{Opt}} + (\Theta^{xy}_{r,r_{Opt}} + \Theta^{+/-}_{r,r_{Opt}}) \cdot L \qquad \forall\, r \neq r_{Opt} \tag{5.8}$$

$$x_{r_{Opt}} + s^x_{r_{Opt}} \cdot l_{r_{Opt}} \leq x_r + (1 + \Theta^{xy}_{r,r_{Opt}} - \Theta^{+/-}_{r,r_{Opt}}) \cdot L \qquad \forall\, r \neq r_{Opt} \tag{5.9}$$

$$y_r + s^y_r \cdot w_r \leq y_{r_{Opt}} + (1 - \Theta^{xy}_{r,r_{Opt}} + \Theta^{+/-}_{r,r_{Opt}}) \cdot W \qquad \forall\, r \neq r_{Opt} \tag{5.10}$$

$$y_{r_{Opt}} + s^y_{r_{Opt}} \cdot w_{r_{Opt}} \leq y_r + (2 - \Theta^{xy}_{r,r_{Opt}} - \Theta^{+/-}_{r,r_{Opt}}) \cdot W \qquad \forall\, r \neq r_{Opt} \tag{5.11}$$

Gleichungen für die Nachbarschaft der Räume untereinander:

$$x_r + s^x_r \cdot l_r \leq x_{r_{Opt}} + (\Theta^{xy}_{r,r_{Opt}} + \Theta^{+/-}_{r,r_{Opt}}) \cdot L \quad \forall\, r \in neighbor_{r_{Opt}} \tag{5.12}$$

$$x_r + s^x_r \cdot l_r \geq x_{r_{Opt}} - (\Theta^{xy}_{r,r_{Opt}} + \Theta^{+/-}_{r,r_{Opt}}) \cdot L \quad \forall\, r \in neighbor_{r_{Opt}} \tag{5.13}$$

$$x_{r_{Opt}} + s^x_{r_{Opt}} \cdot l_{r_{Opt}} \leq x_r + (1 + \Theta^{xy}_{r,r_{Opt}} - \Theta^{+/-}_{r,r_{Opt}}) \cdot L \quad \forall\, r \in neighbor_{r_{Opt}} \tag{5.14}$$

$$x_{r_{Opt}} + s^x_{r_{Opt}} \cdot l_{r_{Opt}} \geq x_r - (1 + \Theta^{xy}_{r,r_{Opt}} - \Theta^{+/-}_{r,r_{Opt}}) \cdot L \quad \forall\, r \in neighbor_{r_{Opt}} \tag{5.15}$$

$$y_r + s^y_r \cdot w_r \leq y_{r_{Opt}} + (1 - \Theta^{xy}_{r,r_{Opt}} + \Theta^{+/-}_{r,r_{Opt}}) \cdot W \quad \forall\, r \in neighbor_{r_{Opt}} \tag{5.16}$$

$$y_r + s^y_r \cdot w_r \geq y_{r_{Opt}} - (1 - \Theta^{xy}_{r,r_{Opt}} + \Theta^{+/-}_{r,r_{Opt}}) \cdot W \quad \forall\, r \in neighbor_{r_{Opt}} \tag{5.17}$$

$$y_{r_{Opt}} + s^y_{r_{Opt}} \cdot w_{r_{Opt}} \leq y_r + (2 - \Theta^{xy}_{r,r_{Opt}} - \Theta^{+/-}_{r,r_{Opt}}) \cdot W \quad \forall\, r \in neighbor_{r_{Opt}} \tag{5.18}$$

$$y_{r_{Opt}} + s^y_{r_{Opt}} \cdot w_{r_{Opt}} \geq y_r - (2 - \Theta^{xy}_{r,r_{Opt}} - \Theta^{+/-}_{r,r_{Opt}}) \cdot W \quad \forall\, r \in neighbor_{r_{Opt}} \tag{5.19}$$

$$x_r + s^x_r \cdot l_r \geq x_{r_{Opt}} - (1 - \Theta^{xy}_{r,r_{Opt}}) \cdot L + length_{Overlap} \quad \forall\, r \in neighbor_{r_{Opt}} \tag{5.20}$$

$$x_{r_{Opt}} + s^x_{r_{Opt}} \cdot l_{r_{Opt}} \geq x_r - (1 - \Theta^{xy}_{r,r_{Opt}}) \cdot L + length_{Overlap} \quad \forall\, r \in neighbor_{r_{Opt}} \tag{5.21}$$

$$y_r + s^y_r \cdot w_r \geq y_{r_{Opt}} - (\Theta^{xy}_{r,r_{Opt}}) \cdot W + length_{Overlap} \quad \forall\, r \in neighbor_{r_{Opt}} \tag{5.22}$$

$$y_{r_{Opt}} + s^y_{r_{Opt}} \cdot w_{r_{Opt}} \geq y_r - (\Theta^{xy}_{r,r_{Opt}}) \cdot W + length_{Overlap} \quad \forall\, r \in neighbor_{r_{Opt}} \tag{5.23}$$

Binäre Variablen für die Nachbarschaft der Räume zu den Außenwänden des Gebäudes:

$$\Lambda^E_r = \begin{cases} 1 & \text{wenn sich der Raum } r \text{ an der Gebäude-Ostseite befindet,} \\ 0 & \text{sonst} \end{cases} \tag{5.24}$$

$$\Lambda^W_r = \begin{cases} 1 & \text{wenn sich der Raum } r \text{ an der Gebäude-Westseite befindet,} \\ 0 & \text{sonst} \end{cases} \tag{5.25}$$

$$\Lambda_r^N = \begin{cases} 1 & \text{wenn sich der Raum } r \text{ an der Gebäude-Nordseite befindet,} \\ 0 & \text{sonst} \end{cases} \tag{5.26}$$

$$\Lambda_r^S = \begin{cases} 1 & \text{wenn sich der Raum } r \text{ an der Gebäude-Südseite befindet,} \\ 0 & \text{sonst} \end{cases} \tag{5.27}$$

Gleichungen für die Nachbarschaft der Räume zu den Außenwänden des Gebäudes:

$$E \in exteriorWall_{r_{Opt}} \quad \Rightarrow \quad x_{r_{Opt}} + s^x_{r_{Opt}} \cdot l_{r_{Opt}} \leq x_E + (1 - \Lambda_r^E) \cdot L \tag{5.28}$$

$$E \in exteriorWall_{r_{Opt}} \quad \Rightarrow \quad x_{r_{Opt}} + s^x_{r_{Opt}} \cdot l_{r_{Opt}} \geq x_E - (1 - \Lambda_r^E) \cdot L \tag{5.29}$$

$$W \in exteriorWall_{r_{Opt}} \quad \Rightarrow \quad x_W \leq x_{r_{Opt}} + (1 - \Lambda_r^W) \cdot L \tag{5.30}$$

$$W \in exteriorWall_{r_{Opt}} \quad \Rightarrow \quad x_W \geq x_{r_{Opt}} - (1 - \Lambda_r^W) \cdot L \tag{5.31}$$

$$N \in exteriorWall_{r_{Opt}} \quad \Rightarrow \quad y_{r_{Opt}} + s^y_{r_{Opt}} \cdot w_{r_{Opt}} \leq y_N + (1 - \Lambda_r^N) \cdot L \tag{5.32}$$

$$N \in exteriorWall_{r_{Opt}} \quad \Rightarrow \quad y_{r_{Opt}} + s^y_{r_{Opt}} \cdot w_{r_{Opt}} \geq y_N - (1 - \Lambda_r^N) \cdot L \tag{5.33}$$

$$S \in exteriorWall_{r_{Opt}} \quad \Rightarrow \quad y_S \leq y_{r_{Opt}} + (1 - \Lambda_r^S) \cdot L \tag{5.34}$$

$$S \in exteriorWall_{r_{Opt}} \quad \Rightarrow \quad y_S \geq y_{r_{Opt}} - (1 - \Lambda_r^S) \cdot L \tag{5.35}$$

$$\sum_{\lambda} \Lambda^{\lambda}_{r_{Opt}} \geq 1 \qquad \lambda \in exteriorWall_{r_{Opt}} \tag{5.36}$$

Für eine sinnvolle Dimensionierung der Räume werden noch zusätzliche Ungleichungen zur Einhaltung eines minimalen und maximalen Verhältnisses der Breite und Länge zueinander vorgegeben. Dafür werden keine Funktionalitäten des Packingframeworks verwendet, statt dessen werden diese als eigenständige Nebenbedingungen formuliert, welche das Gesamtproblem erweitern. Der Wert für den Verhältnisfaktor kann für jeden Raum aus dem Entwurfsgraphen ausgelesen werden, da er zuvor in den Regeln der Entwurfssprache vorgegeben wurde.

Zusätzliche Gleichungen für das zulässige Verhältnis der Raumlänge zur Raumbreite:

$$s^x_{r_{Calc}} - \frac{s^y_{r_{Calc}}}{F} \geq 0 \tag{5.37}$$

$$s^x_{r_{Calc}} - s^y_{r_{Calc}} \cdot F \leq 0 \tag{5.38}$$

Abschließend wird die Zielfunktion definiert, welche die Flächen aller Wohnungsräumlichkeiten ausgenommen der Flure maximieren soll. Das Auslassen der Flure hat den Hintergrund, dass ausschließlich die tatsächliche Nutzfläche maximiert wird. Zur Maximierung der Gesamtfläche wird für jeden zu maximierenden Wohnraum r_{Max} die `RectangularArea` Klasse des Packingframeworks als Kind der `ObjectiveBuilder` Klasse aufgerufen. Um nun alle einzelnen Zielfunktionen zu vereinen, also statt der einzelnen Raumflächen die Gesamtfläche zu maximieren, kann die `ObjectiveAggregator` Klasse des Packingframeworks verwendet werden. Diese bildet aus allen übergebenen Zielen eine globale, alle Einzelziele repräsentierende Zielfunktion.

Zielfunktion:

$$max \sum_{r_{Max}=0}^{M_{Max}} s^x_{r_{Max}} \cdot l_{r_{Max}} \cdot s^y_{r_{Max}} \cdot w_{r_{Max}} \tag{5.39}$$

Es wird deutlich, dass nur wenige Schritte bei der Programmierung des eigentlichen Optimierungsmodells tatsächlich von dem/der Programmierer/-in übernommen werden müssen und die meisten der Teilmodelle durch den Einsatz der Methoden aus dem Packingframework für die manuelle Implementierung entfallen. Der zusätzliche Definitionsaufwand umfasst die einmalige Kodierung der Instanziierung der Freiheitsgradvariablen und die Formulierung der Zusatzbedingung des räumlichen Verhältnisses, welche dem Gesamtmodell zusätzlich hinzugefügt wird. Die übrigen Zusatzvariablen und Ungleichungen, welche sich aus den Adjazenzen, den Inklusionen und der Kollisionsfreiheit ergeben sowie die Repräsentation der globalen Zielfunktion samt der Teilziele werden automatisiert erstellt.

Optimierung für ein freistehendes Gebäude

Die erste Fragestellung befasst sich mit der Layoutoptimierung zweier Wohnungen in einem der freistehenden Gebäudeblöcke. Zur Repräsentation dieses Anwendungsfalls wird vorgegeben, dass jeder der Wohnräume einen beliebigen Außenwandkontakt haben muss. Diese Vorgabe erfolgt durch die Vorgabe des Enumerationswertes OutsideContact:ANY. Dieser Wert hat zur Folge, dass die `OrBuilder` Klasse alle Gebäudeseiten als mögliche Kontaktseite berücksichtigt.

Die Optimierung liefert nach 49.44 Sekunden eine global optimale Lösung von 168.917 Quadratmetern für die Wohnräume beider Wohnungen zusammen. Abb. 5.16 zeigt die Ergebnisse des Anwendungsfalls. Alle vorgegebenen Räume, wie das Treppenhaus und die Balkone der Wohnungen, sind hierbei grau eingefärbt. Die beiden Wohnungen sind durch unterschiedliche Blauschattierungen zu unterscheiden. Die Ergebnisse sind eingebettet im Lösungsprozess abgebildet. Beginnend bei der letztlich berechneten optimalen Lösung oben in der Mitte, werden ausgewählte valide Zwischenschritte, also solche, die alle Nebenbedingungen erfüllen, während der Optimierung rückwärts abgebildet. Zu sehen sind sowohl die topologischen wie auch die rein parametrischen Änderungen. Diese sind in den einzelnen Layoutlösungen auch separat über Symbole und Farben immer im Vergleich zum Nachfolger gekennzeichnet. Sind die Bezeichnungen der Räume weiß, hat keine (oder eine vernachlässigbar geringe) Änderung stattgefunden. Die schwarze Färbung und das Symbol, welches ein bemaßtes Rechteck darstellt, symbolisieren eine rein parametrische Änderung. Eine rote Färbung sowie ein Doppelpfeil symbolisieren eine topologische Änderung während der Lösung. Insgesamt ist zu erkennen, dass die Flurgröße in der optimalen Lösung geringer ist als im ersten dargestellten Zwischenschritt. Weiterhin lässt sich die globale Konvergenz des Lösungsprozesses beobachten, welche allerdings aufgrund topologischer Änderungen lokale Divergenzen aufweist. Besonders zwischen Schritt fünf und sechs fällt hierbei der Anstieg der Anzahl der Änderungen auf, was lediglich eine qualitative Beobachtung darstellt, da aufgrund einer fehlenden Maßzahl für die Änderung keine quantitativen Aussagen gemacht werden können.

Optimierung eines Reihenendblocks im Gebäudekomplex

Für eine zweite Anwendung soll nun dieselbe Layoutoptimierung für ein Stockwerk in einem Block aus einem Doppelgebäude durchgeführt werden. Hierfür werden die Regeln der Entwurfssprache so verändert, dass nur noch Kontakte zwischen Räumen und dreien der vier

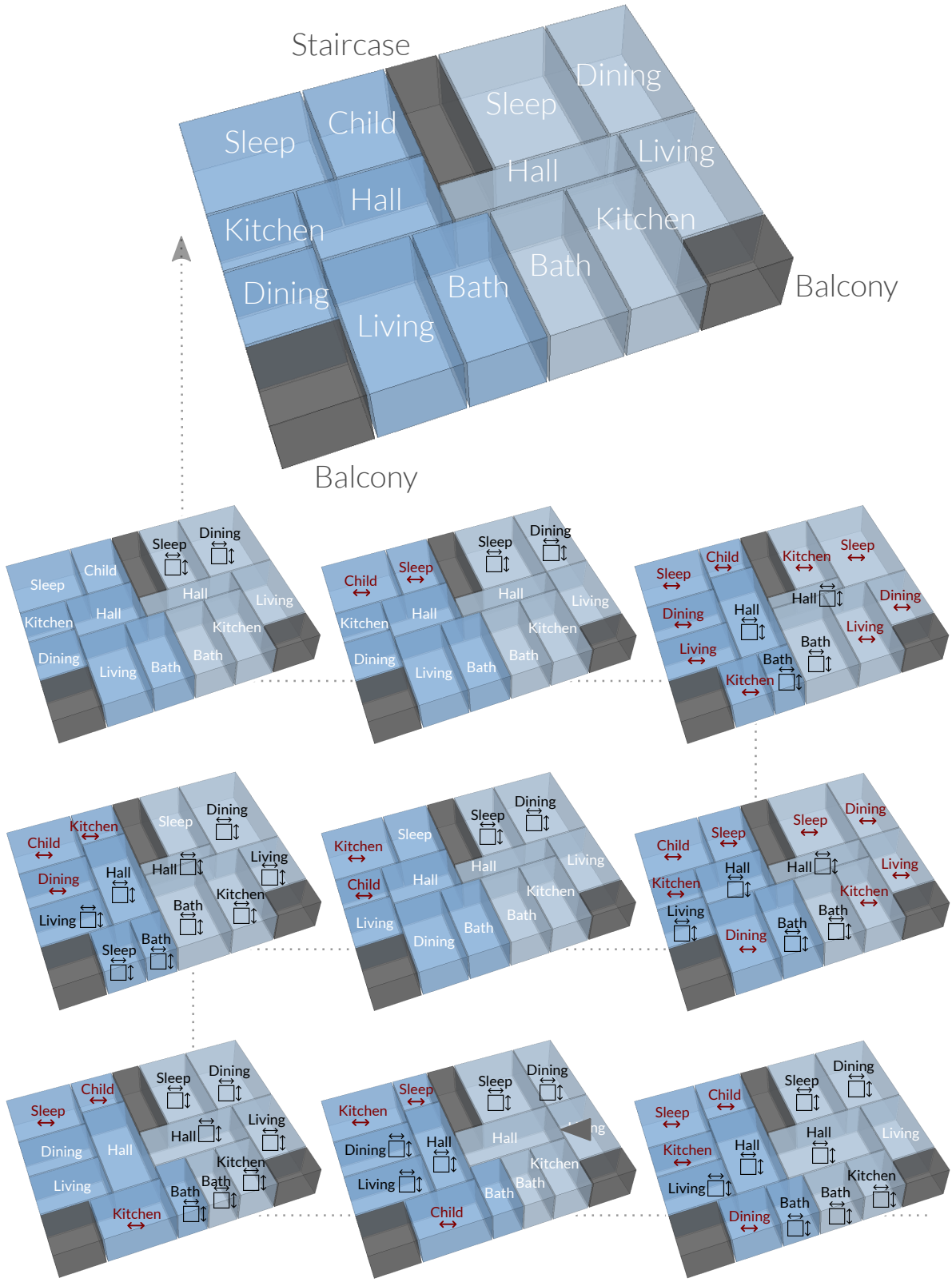


Abbildung 5.16: Lösung der Layoutoptimierung mit Zwischenlösungen während der Optimierung. Färbung und Zeichen symbolisieren einen Wechsel des Vorgängers zu seinem Nachfolger. Weiß: Keine Änderung. Schwarz: Parametrische Änderung. Dunkelrot: Topologische Änderung.

Außenwände zulässig sind. Im konkreten Fall wird für den Außenkontakt der Enumerationswert.NSW festgelegt. Wird die Optimierung ohne Änderung erneut durchgeführt, wird das Problem als unlösbar erkannt. Daraus lässt sich für den/die Anwender/-in schließen, dass keine valide Lösung existiert. Als Reaktion darauf können die Restriktionen gelockert werden. Es wird also zusätzlich beschlossen, dass auch die Balkone als Außenwand fungieren können und (für eine Vorauslegung) dennoch eine akzeptable Ausleuchtung gewährleistet ist. Die Balkone dienen nun nicht mehr ausschließlich als Hindernisse bei der Kollisionsvermeidung, sondern müssen auch als potenzielle Nachbarn der Wohnräume berücksichtigt werden. Infolgedessen wird die Klasse `Predefined` des Klassendiagramms aus Abb. 5.13 um das boolesche Attribut `isOutside` erweitert. Damit kann spezifiziert werden, welche Räumlichkeiten als Außenkontakt gelten. Das Gesamtproblem kann mithilfe der `ContactBuilder` Klasse um die möglichen Kontakte der Räume zu den Balkonen erweitert werden. An die `OrBuilder` Klasse müssen außerdem zusätzlich die Balkon-Kontakt-Bedingungen übergeben werden (siehe Gl. (5.52)).

Ergänzender Index:

b: Index für den Raum $b \in M_{Balkony}$ zur Optimierung

Ergänzender Parameter:

$M_{Balkony}$: Menge der Räume, welche Balkone sind

Ergänzende Binäre Variablen für die relative Positionierung der Räume zu den Außenwänden des Gebäudes oder der Balkone:

$$\Lambda^E_{r,b} = \begin{cases} 1 & \text{wenn sich der Raum } r \text{ zur Ostseite am Balkon } b \text{ befindet,} \\ 0 & \text{sonst} \end{cases} \tag{5.40}$$

$$\Lambda^W_{r,b} = \begin{cases} 1 & \text{wenn sich der Raum } r \text{ zur Westseite am Balkon } b \text{ befindet,} \\ 0 & \text{sonst} \end{cases} \tag{5.41}$$

$$\Lambda^N_{r,b} = \begin{cases} 1 & \text{wenn sich der Raum } r \text{ zur Nordseite am Balkon } b \text{ befindet,} \\ 0 & \text{sonst} \end{cases} \tag{5.42}$$

$$\Lambda^S_{r,b} = \begin{cases} 1 & \text{wenn sich der Raum } r \text{ zur Südseite am Balkon } b \text{ befindet,} \\ 0 & \text{sonst} \end{cases} \tag{5.43}$$

Ergänzende Gleichungen und Änderungen für die Nachbarschaft der Räume zu den Außenwänden des Gebäudes oder der Balkone:

$$E \in extW_{r_{Opt}} \Rightarrow x_{r_{Opt}} + s^x_{r_{Opt}} \cdot l_{r_{Opt}} \leq x_b + (1 - \Lambda^E_{r,b}) \cdot L \quad \forall \{b \mid x_b + s^x_b \cdot l_b = x_E\} \tag{5.44}$$

$$E \in extW_{r_{Opt}} \Rightarrow x_{r_{Opt}} + s^x_{r_{Opt}} \cdot l_{r_{Opt}} \geq x_E - (1 - \Lambda^E_{r,b}) \cdot L \quad \forall \{b \mid x_b + l_b = x_E\} \tag{5.45}$$

$$W \in extW_{r_{Opt}} \Rightarrow x_b + l_b \leq x_{r_{Opt}} + (1 - \Lambda^W_{r,b}) \cdot L \quad \forall \{b \mid x_b = x_W\} \tag{5.46}$$

$$W \in extW_{r_{Opt}} \Rightarrow x_b + l_b \geq x_{r_{Opt}} - (1 - \Lambda^W_{r,b}) \cdot L \quad \forall \{b \mid x_b = x_W\} \tag{5.47}$$

$$N \in extW_{r_{Opt}} \Rightarrow y_{r_{Opt}} + s^y_{r_{Opt}} \cdot w_{r_{Opt}} \leq y_b + (1 - \Lambda^N_r) \cdot L \quad \forall \{b \mid y_b + w_b = y_N\} \tag{5.48}$$

$$N \in extW_{r_{Opt}} \Rightarrow y_{r_{Opt}} + s^y_{r_{Opt}} \cdot w_{r_{Opt}} \geq y_b - (1 - \Lambda^N_r) \cdot L \quad \forall \{b \mid y_b + w_b = y_N\} \tag{5.49}$$

$$S \in extW_{r_{Opt}} \Rightarrow y_b + w_b \leq y_{r_{Opt}} + (1 - \Lambda^S_r) \cdot L \quad \forall \{b \mid y_b = y_S\} \tag{5.50}$$

$$S \in extW_{r_{Opt}} \Rightarrow y_b + w_b \geq y_{r_{Opt}} - (1 - \Lambda_r^S) \cdot L \qquad \forall \{b \mid y_b = y_S\} \quad (5.51)$$

$$\sum_{\lambda} \Lambda^{\lambda}_{r_{Opt}} + \Lambda^{\lambda}_{r_{Opt},b} \geq 1 \qquad \lambda \in extW_{r_{Opt}} \quad (5.52)$$

Mit den genannten Erweiterungen kann bei der Optimierung des Problems nach 70.23 Sekunden eine global optimale Lösung von 168.917 Quadratmetern für die Wohnräume beider Wohnungen zusammen gefunden werden. Zwar weist diese Lösung keine Verbesserung gegenüber der Lösung aus Abschnitt 5.1.2 auf (für beide Problemstellungen liefert die Optimierung genau dasselbe Layout als global bestes Ergebnis), trotz der erhöhten Zwänge aufgrund einer fehlenden Außenwand kann allerdings immerhin ein gleichwertiges Ergebnis gefunden werden.

Optimierung eines Reihenmittelblocks im Gebäudekomplex

Im letzten Anwendungsfall des Beispiels soll nun noch einen Schritt weiter gegangen werden und die Layoutoptimierung für einen mittleren Block eines dreifachen Gebäudekomplexes durchgeführt werden. Für diesen Fall wird in den Regeln der Entwurfssprache für den Außenkontakt der Enumerationswert.NS festgelegt. Auch diese Problemstellung liefert trotz Lockerung der Außenwandbedingungen die Unlösbarkeit als Ausgabe. Es kann also erneut geschlossen werden, dass für eine solche Problemstellung keine Lösung existiert. Da weder die Minimal- und Maximaldimensionen der Räume, noch die der Verhältnisfaktoren verändert werden sollen, wird nun beschlossen, die Treppenhausposition an der Nordwand freizugeben. Dafür wird lediglich in den graphischen Regeln der Entwurfssprache die *Predefined* Klasse zur Repräsentation des Treppenhauses gegen eine `RoomToPlace` Klasse getauscht. Außerdem wird für den Außenkontakt des Treppenhauses der Enumerationswert.N festgelegt. Die Optimierung erzielt nach 594.68 Sekunden ein global optimales Ergebnis von 177.5 Quadratmetern für die Wohnräume beider Wohnungen zusammen. Durch die Lockerung der Positionszwänge des Treppenhauses kann somit trotz der zusätzlichen Zwänge durch zwei fehlende Außenwände ein noch besseres Ergebnis erzielt werden als bei den beiden vorherigen Anwendungsfällen. Das berechnete Ergebnis kann Abb. 5.17 entnommen werden.

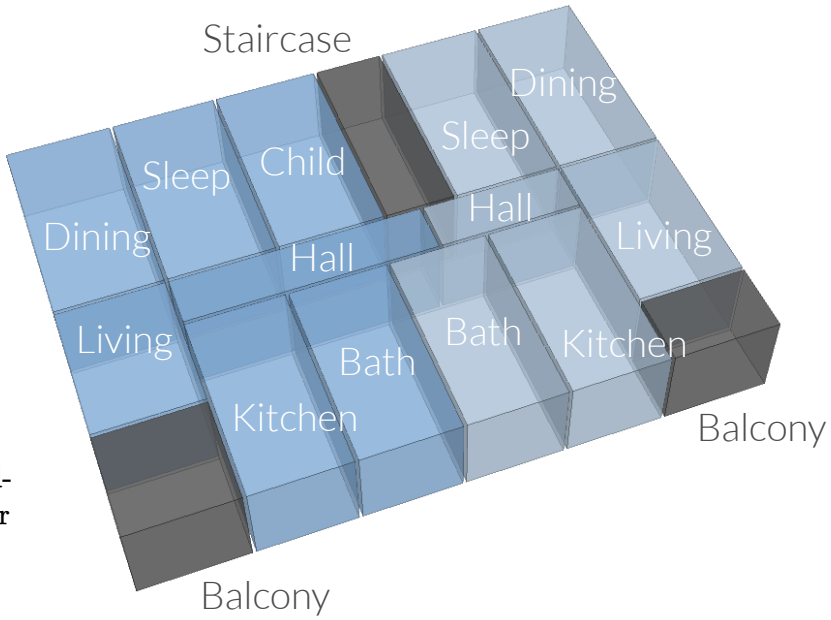


Abbildung 5.17: Lösung für die Optimierung der Grundrisse zweier Wohnungen eines Reihenmittelhauses mit freier Treppenhausposition an der Nordwand. Positionierung der Wohnräume an Nord- oder Süd-Wänden für den Zugang zu Tageslicht.

5.2 Packing in einem Flugzeugtragflügel

Im vorliegenden Kapitel wird die Entwicklung und Implementierung einer Softwareplattform vorgestellt, mithilfe derer der modellbasierte Systementwicklungsprozess auf Basis der *Model Based Systems Engineering* Vorgehensweise automatisiert werden kann. Ziel der Entwicklung ist, eine Softwarelösung zu schaffen, welche physische Systemarchitekturen automatisiert und iterativ generiert und damit deren Optimierung unterstützt. Die erfolgreiche Umsetzung kann am Beispiel der optimalen Auslegung eines Flugzeugtragflügels demonstriert werden und wird in der vorliegenden Arbeit mit Fokus auf dem Konfigurations- bzw. Packingprozess besprochen. Es sei an dieser Stelle darauf hingewiesen, dass der vorliegende Abschnitt hauptsächlich auf Teilen der Ergebnisse des *CLEANSKY2*-Forschungsprojekts *PHAROS (Physical Architecture Optimization System)* [Rudolph, 2019][CleanSky2, 2022][CleanAviation Media, 2022] basiert. Das konkrete Ziel des Projektes ist die Systemintegration der bereits separat bestehenden Entwurfssprachen für *Packing*, *Piping* (Verrohrung) und *Routing* (Verkabelung). Für die Automatisierung der ehemals manuellen Prozesskette aus aufeinander folgenden Entwurfssprachen wird ein sogenannter *Software-Stack* bestehend aus dem *Design Cockpit 43*®und der Optimierungssoftware Optimus®[Noesis Solutions, 2022] aufgebaut. Der daraus resultierende Gesamtprozess kann Abb. 5.18 entnommen werden.

In der vorliegenden Arbeit liegt der Fokus auf dem Packingverfahren. Die Verrohrung und Verkabelung sind dabei nicht Teil der vorliegenden Arbeit und finden mithilfe speziell dafür entwickelter und zum Zeitpunkt der Implementierung der Beispiele bereits vorhandener Entwurfssprachen statt. Der interessierte Leser sei für eine detaillierte Beschreibung des Gesamtprojektes auf [Dinkelacker et al., 2021] verwiesen.

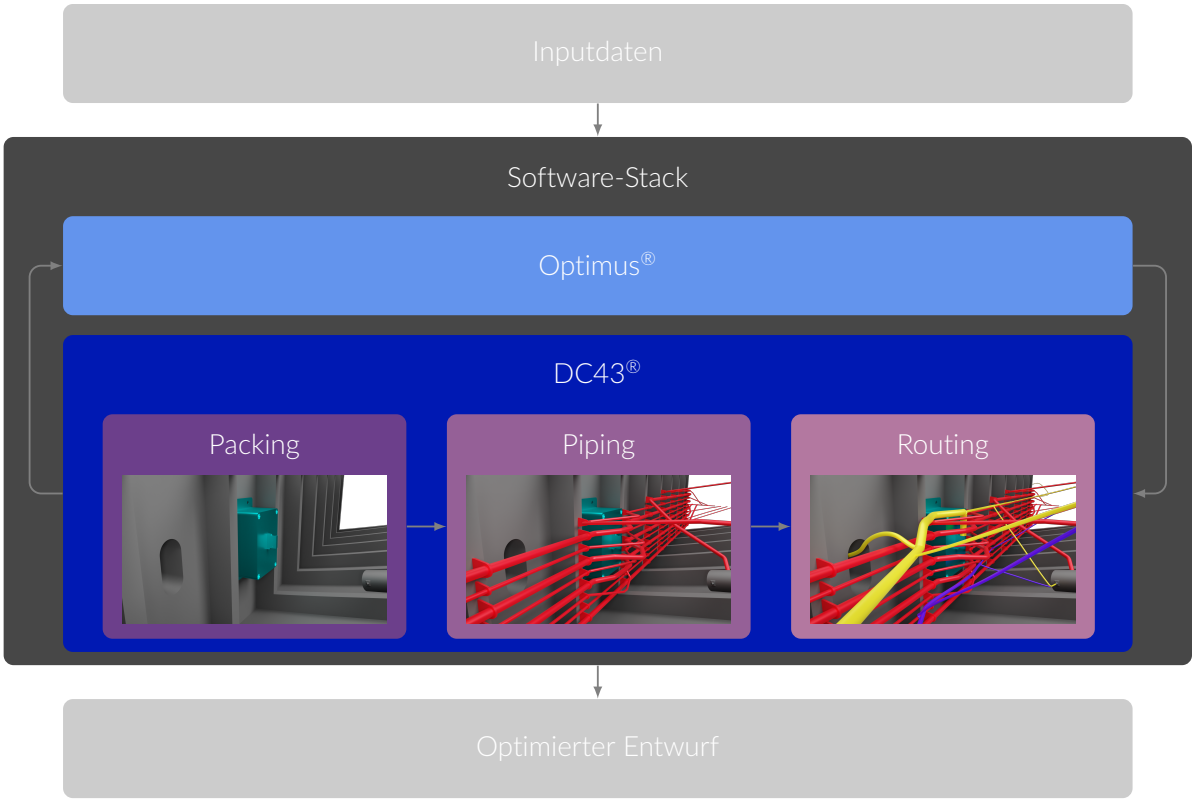


Abbildung 5.18: Software-Stack [Dinkelacker et al., 2021].

5.2.1 Packing als Teil der Gesamtsystemintegration

Um die Umsetzung des Packings als Teil der Gesamtsystemintegration zu erläutern, soll nochmals genauer auf Abb. 5.18 eingegangen werden. Der Aufbau des gesamten Software-Stacks ist so gestaltet, dass das *Design Cockpit 43*®iterativ aus Optimus®aufgerufen wird, sodass in jeder Schleife alle drei Entwurfssprachen automatisiert ausgeführt werden können. Die Ergebnisse der Entwurfssprachen werden am Ende jeder Iteration wieder in Optimus®eingelesen, wodurch die relevanten initialen Modellparameter für die folgende Iteration modifiziert werden können. Mit diesem Vorgehen kann also eine Optimierung über alle theoretisch möglichen Systemvarianten, im vorgestellten Anwendungsfall am Beispiel eines Tragflügels, durchgeführt werden. Sobald das *Design Cockpit 43*®aufgerufen wird, beginnt die Auslegung der Tragflügelkonfiguration mithilfe der Packing-Entwurfssprache. Diese positioniert relevante Bauteile unter Berücksichtigung der durch die Optimierung vorgegebenen Entwurfsparameter und überprüft deren Kollisionsfreiheit. Darauf folgen die Prozesse für die Verrohrung und die Verkabelung. Die Prozesskette ist dabei so modelliert, dass eine invalide Lösung frühzeitig erkannt wird, um rechenintensive Verrohrungs- und Verkabelungsprozeduren für invalide Lösungen zu vermeiden. Innerhalb des Packingsprozesses wird beispielsweise eine Lösung direkt als ungültig gewertet, sobald eine Kollision erkannt wird. Ist dies der Fall, wird die gesamte Optimierungsschleife unterbrochen und satt dessen wird direkt im Anschluss die nächste Iteration mit neuen initialen Variablenwerten ausgeführt. Zur Validierung des Software-Stacks wird eine ganzheitliche Betrachtung von Tragflügelkonfigurationen vorgenommen, in der aussagekräftige Auslegungsparameter aus der Tragflügelplanung ausgewertet und optimiert werden. Für den vorgestellten Anwendungsfall werden die folgenden Kriterien näher betrachtet:

- die gesamte Kabelbaumlänge
- die gesamte Verrohrungslänge
- die Anzahl der Biegungen der Verrohrung

Der Packingprozess im Gesamtsystem beginnt mit der Verarbeitung der Bauraumgeometrie inklusive der Flügelstruktur, der Flügelhaut sowie der Aktuatoren und Sensoren aus einem zuvor eingelesenen Modell. Für eine einfachere Verarbeitung wird die Flügelstruktur in einzelne Kammern geteilt. Diese werden fortlaufend nummeriert, wie in Abb. 5.19 zu entnehmen ist, sodass sich am Vorderflügel die Kammern 0 bis 47 befinden und an der Rückseite die Kammern 48 bis 88. Die erste Aufgabe des Packingverfahrens ist nun eine geeignete Flügeltopologie für die folgende Verkabelung bereitzustellen. Im vorgestellten Anwendungsfall werden drei mögliche Hauptrouten definiert. Die topologische Variation ergibt sich aus dem Öffnen oder Schließen der Routen. In Abb. 5.19 sind alle möglichen kombinatorischen Varianten dargestellt. Für die Öffnung oder Schließung wird pro Route eine boolesche Optimierungsvariable eingeführt, sodass sich insgesamt drei boolesche Variablen ergeben. Außerdem ist das Packingverfahren für die geometrische Ausprägung dieser Randbedingung zuständig, sodass die Route für die Verkabelung zugänglich oder nicht zugänglich ist.

Eine weitere Aufgabe des Packingverfahrens ist die Verteilung von elektronischen Verteilerboxen innerhalb des Flügels, welche als Ausgangsbasis für die Verkabelung dienen und gleichzeitig Hindernisse für die Verrohrung darstellen. Im beschriebenen Anwendungsfall handelt es sich hierbei um den Verteiler der Steuerungseinheit für die Aktuatoren (*Actuator Control Unit (ACU)*) und die Verteiler von zwei Generatoren. Da die Gesamtkomplexität der aufeinander aufbauenden Aufgaben aus Packing, Verrohrung und Verkabelung enorm ansteigt, ist es notwendig, den Entwurfsraum bereits zu Anfang stark einzugrenzen. Hierfür wird die Kammeraufteilung genutzt. Aus der Nummerierung der Flügelkammern ergeben sich diskrete

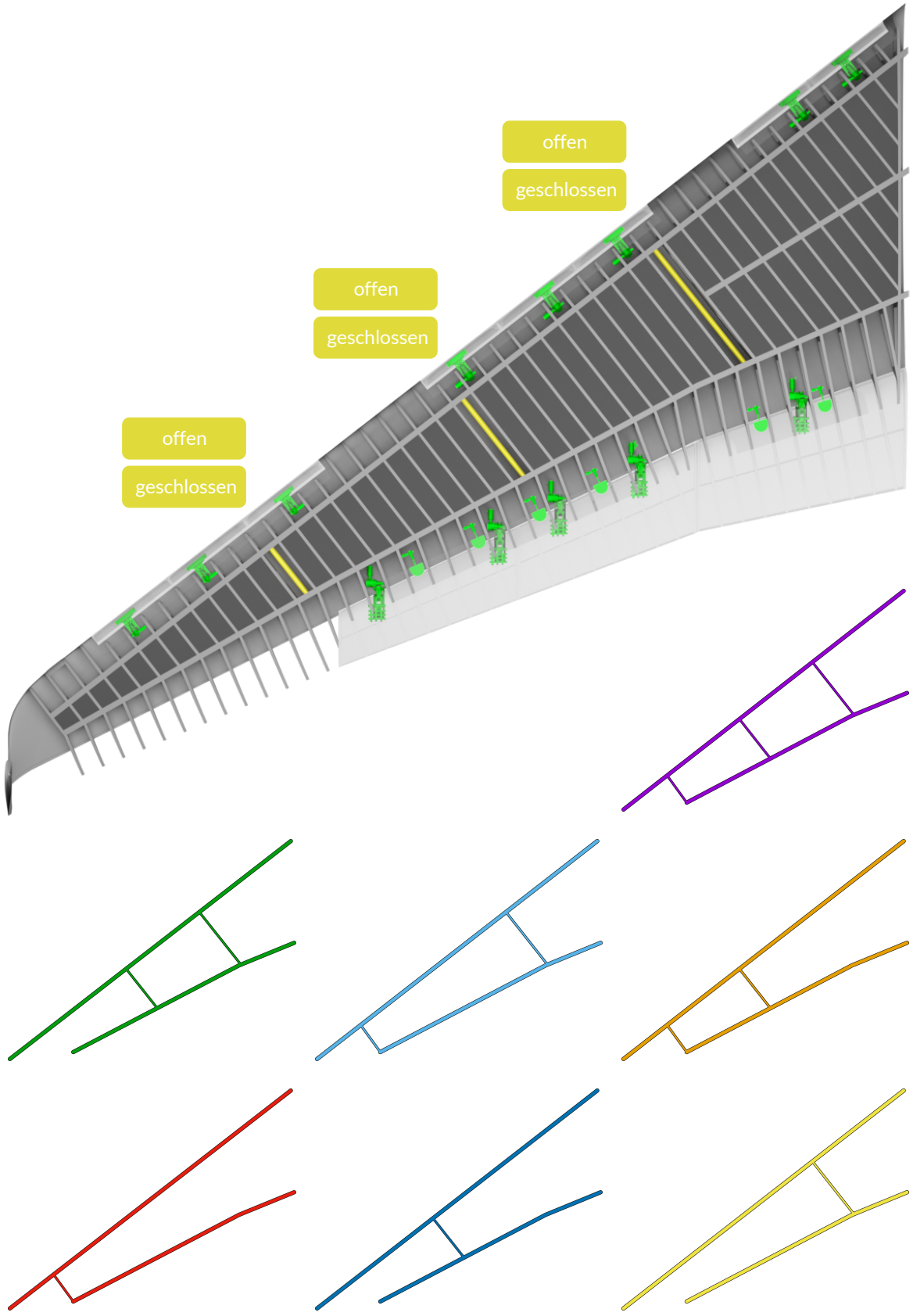


Abbildung 5.19: Diskrete Zustände der Kabelkorridore und die sich daraus ergebenden Verkabelungskonfigurationen [Dinkelacker et al., 2021].

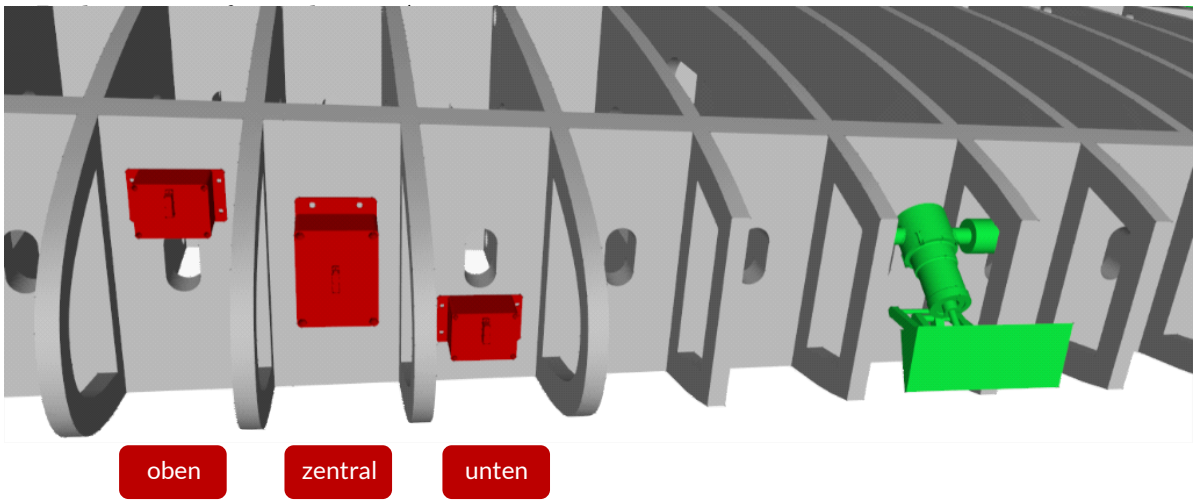


Abbildung 5.20: Diskrete Positionen der Elektronikkomponenten wie Anschlussboxen für die Generatoren und die ACU [Dinkelacker et al., 2021].

Freiheitsgrade für die elektronischen Boxen bezüglich ihrer x- und z-Koordinate. Die Diskretisierung führt zur späteren Laufzeitersparnis, da nur noch für eine vorgegebene Anzahl an diskreten Konfigurationslösungen automatisiert verrohrt und verkabelt werden muss. Weiterhin kann hierdurch auf eine zusätzliche rechenaufwändige Kollisionserkennung verzichtet werden, da die Kollisionsfreiheit der Boxen untereinander sowie mit der Flügelstruktur implizit enthalten ist. Es bleiben lediglich die Kollisionen mit der Flügelhaut und den Aktuatoren sowie Sensoren zu überprüfen und zu vermeiden. Die y-Koordinate der elektronischen Boxen wird ebenfalls als diskreter Wert angegeben, welcher die folgenden Werte annehmen kann:

- *oben*: untere Kante der Flügelstruktur + 0.75 · Strukturhöhe
- *zentral*: untere Kante der Flügelstruktur + 0.5 · Strukturhöhe
- *unten*: untere Kante der Flügelstruktur + 0.25 · Strukturhöhe

Werden alle genannten Variablen berücksichtigt, ergibt sich alleine für das Packing eine theoretische Variation von: $88 \cdot 87 \cdot 86 \cdot 3 \cdot 3 \cdot 3 \cdot 7 = 124.440.624$ Varianten, da

- jede der elektronischen Boxen innerhalb einer der 88 Kammern positioniert werden kann,
- jede der elektronischen Boxen vertikal eine aus 3 Positionen annehmen kann,
- 7 Varianten für die Topologie der Hauptrouten existieren.

Für eine weitere Laufzeitreduktion werden weitere Einschränkungen getroffen. Zunächst werden die x-z-Positionen der elektronischen Komponenten eingeschränkt. Während der Verteiler für die Steuereinheit weiterhin in jeder Kammer positioniert werden kann, wird der erste Generatorverteiler nun ausschließlich an der Flügelrückseite positioniert und der zweite Generatorverteiler ausschließlich an der Flügelvorderseite. Weiterhin wird der Freiheitsgrad zur Positionierung in y-Richtung auf eine statt drei möglichen Positionen beschränkt. Damit ergibt sich eine reduzierte theoretische Variation von: $88 \cdot 47 \cdot 41 \cdot 1 \cdot 1 \cdot 1 \cdot 7 = 1.187.032$ Varianten, da

- der Verteiler für die Steuerungseinheit innerhalb einer der 88 Kammern, der erste Generatorverteiler innerhalb der 47 vorderen und der zweite Generatorverteiler innerhalb der 41 hinteren Kammern positioniert werden kann,
- jede der elektronischen Boxen vertikal genau eine Position annehmen kann,
- 7 Varianten für die Topologie der Hauptrouten existieren.

Die berechnete Komplexität wird als akzeptabel gewertet und dient als Basis für den Gesamtprozess. Demnach gehen in den Packingprozess 1.187.032 Varianten ein, welche auf etwa 40.000 tatsächlich valide Lösungen zusammenschrumpfen, für welche eine vereinfachte Verrohrung stattfindet. Daraus werden die vielversprechendsten Lösungen herausgefiltert, welche im weiteren Verlauf verrohrt werden. In Abb. 5.21 können die Auswertungen der vollständig durchlaufenen Varianten innerhalb der Optimierung eingesehen werden. In den Schaubildern werden die Ergebnisse in Abhängigkeit der Position des Verteilers der Steuerungseinheit und der Generatorverteilerpositionen aufgetragen. Die x-Koordinate stellt die jeweilige Komponentenposition dar, die y-Koordinate repräsentiert den Wert für die Gesamtlänge des Kabelbaums, die Färbung repräsentiert die Gesamtlänge der Verrohrung und der Durchmesser der kreisförmig dargestellten Messpunkte gibt Auskunft über die Anzahl der Rohrbiegungen, wobei ein wachsender Durchmesser eine wachsende Anzahl an Biegungen darstellt. Aus diesen Auswertungen können nun Varianten von dem/der Ingenieur/-in ausgewählt werden, welche geeignet erscheinen. In Abbildung 5.22 ist eine der im Projekt ausgewählten Varianten dargestellt.

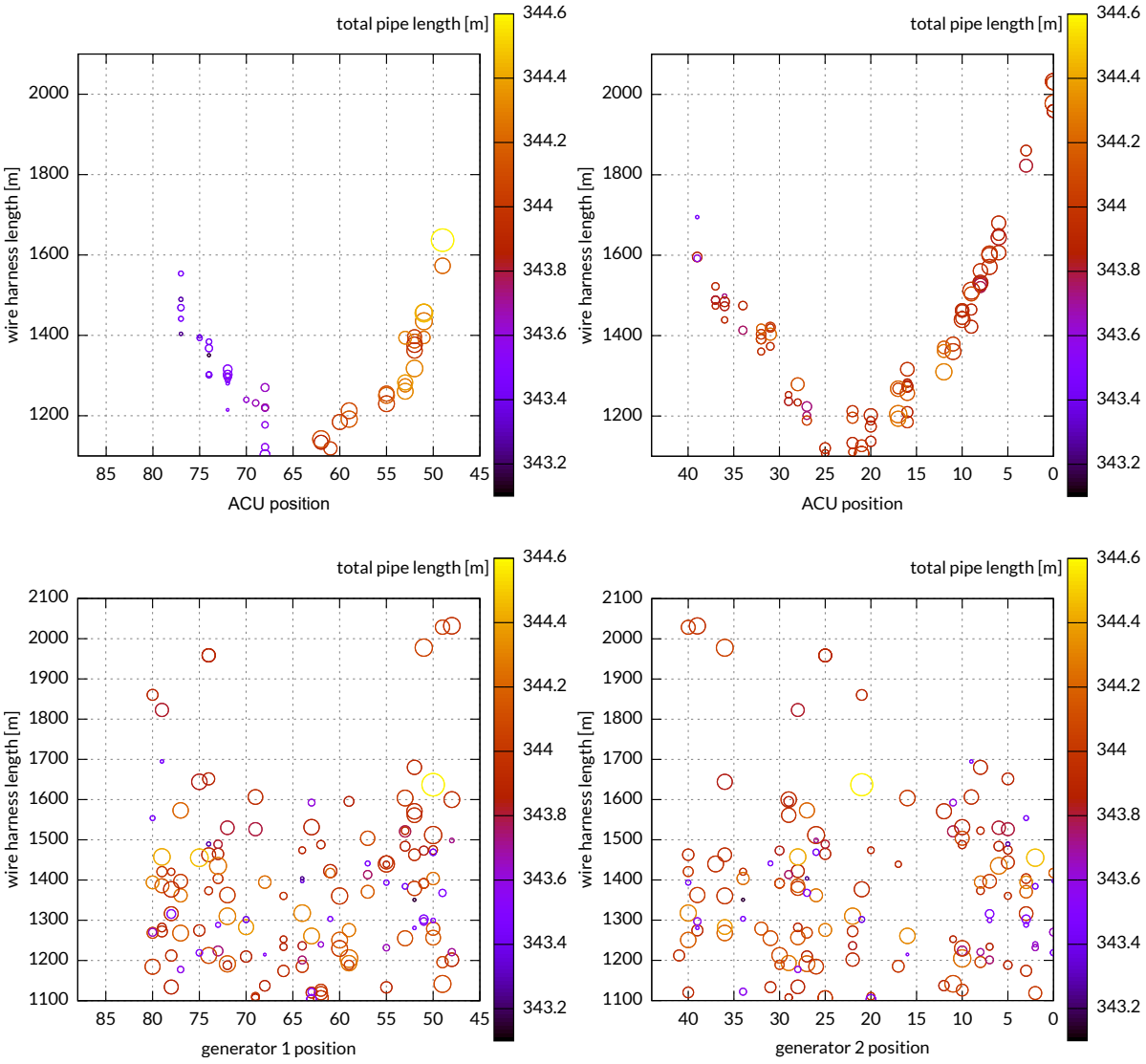


Abbildung 5.21: Auswertung der Optimierungsergebnisse [Dinkelacker et al., 2021].

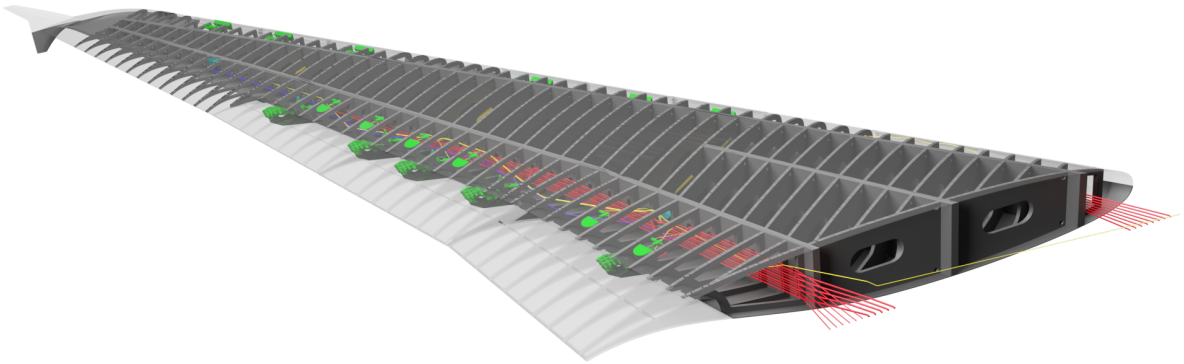


Abbildung 5.22: Gesamtergebnis [Dinkelacker et al., 2021].

5.2.2 Metaheuristische Positionierung von Komponenten

Für die Gesamtsystemintegration innerhalb des Software-Stacks wurde die packingspezifische Fragestellung stark vereinfacht. Im vorliegenden Abschnitt soll eine umfangreichere Aufgabenstellung betrachtet werden, in der die Positionierung einer größeren Anzahl von elektronischen Verteilerkomponenten mithilfe der Partikel-Multi-Schwarm-Optimierung aus dem entwickelten Packingframework durchgeführt wird. Hierbei sollen die folgenden Komponenten im bestehenden Tragflügel positioniert werden:

- Transformatoren für 8 Vorflügel-Aktuatoren
- Transformatoren für 5 Landeklappen-Aktuatoren
- Transformatoren für 6 Störklappen-Aktuatoren
- je einen Hauptverteilerbus für 2 Generatoren (Nähe Triebwerke)
- je einen Stromrichter für 2 Generatoren (Nähe Triebwerke)
- Hauptverteilerbus für die ACU
- je zwei redundante Verteiler zu den Transformatoren desselben Typs
- zwei redundante Zwischenverteiler vom ACU-Bus zu den genannten Verteilern

Die Vernetzung der Komponenten untereinander ist in Abb. 5.23 dargestellt. Ziel des Packingverfahrens ist mithilfe einer einfachen Kabellängenabschätzung die Boxen so zu positionieren, dass die Gesamt-Kabelmenge minimiert wird. Die Zielfunktion umfasst dafür die Minimierung der einzelnen adjazenten Distanzen zwischen den Komponenten. Für die Kabellängenabschätzung zwischen den elektronischen Verteilern werden alle drei Passagen als geöffnet angenommen. Die Berechnung der Kabellängen findet über eine Summation der euklidischen Abstände der Boxen und der adjazenten Komponenten zu den Eingangs- und Ausgangspunkten der Passagen und der Länge der jeweiligen Passage statt. Es geht immer die kürzeste aller kombinatorisch möglichen Varianten in die Auswertung der Zielfunktion ein. Die Entfernungen zwischen den Aktuatoren und zugehörigen Transformatoren werden direkt über die euklidische Distanz berechnet, dasselbe gilt für die Stromrichter und Hauptverteiler der Generatoren, deren Abstand zur vorgegebenen Generatorposition gemessen wird. Die Gewichtung der einzelnen Strecken orientiert sich an einer Abschätzung der Kabelanzahl zwischen den jeweiligen

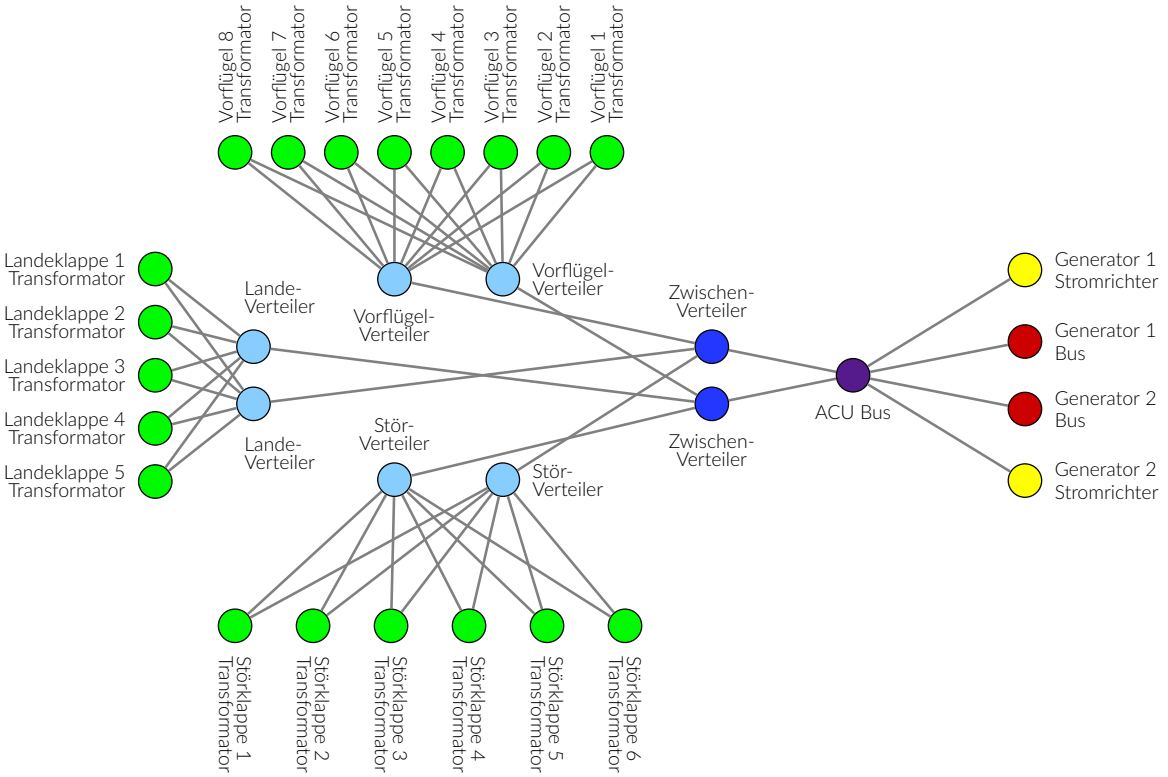


Abbildung 5.23: Vernetzung der zu verteilenden elektronischen Komponenten.

Komponenten. Die Nebenbedingungen umfassen die Kollisionsfreiheit aller Boxen zur Struktur und Flügelhaut sowie den Aktuatoren und untereinander. Weitere Nebenbedingungen nutzen die Positionierung an der Strukturwand aus und koppeln die x- und z-Koordinate, sodass nur die y-Position der Komponenten unabhängig gewählt werden kann. Diese wird durch weitere Restriktionen in Abhängigkeit von der x- bzw. z-Position nach oben und unten beschränkt. Für die Boxen, welche in direktem Zusammenhang mit einem Aktuator oder Generator stehen, wird eine zuvor spezifizierte Anzahl an Kammern vorgeben, um welche von der Kammer des zugehörigen Bauteils abgewichen werden kann. Es wurde eine Untersuchungsreihe mit verschiedenen Initialwerten für die Partikel-Multi-Schwarm-Optimierung durchgeführt, welche Tabelle 5.1 entnommen werden können. Für die Berücksichtigung des stochastischen Verhaltens wurde jede Variante zehnmal ausgeführt, wobei nur die erfolgreichste in der Tabelle aufgenommen wurde. Die Tabelle gibt neben den Optimierungsparametern auch Auskunft darüber, ob ein Ergebnis erfolgreich ohne Restriktionsverletzung gefunden werden konnte und enthält die berechneten Werte für die jeweilige Zielfunktion im Erfolgsfall. Für alle Berechnungen gelten die Gewichtungsfaktoren $w_{Individual} = 1.49445$, $w_{Social} = 1.49445$ und $w_{Global} = 0.3645$, die Suchstrategie MCPSOIS, die Trägheit $f = 0.729$ sowie die Straffunktion nach `EpsilonSum` und der Umgang mit Entwurfsraumgrenzen mithilfe von `REFLECTzero`.

Während der Versuchsreihe ist aufgefallen, dass sich die Kollisionsdetektion der zu verteilenden Komponenten mit der Flügelhaut negativ auf das Konvergenzverhalten der Optimierung auswirkt. Dies wird dadurch erklärt, dass die Abweichung der y-Positionen der Komponenten von den integrierten oberen und unteren Schranken kontinuierlich berechnet werden können, während eine Kollision mit der Flügelhaut zu Sprüngen in der Auswertung der Restriktionsverletzungen führt. Dies kann dazu führen, dass eine Komponente nicht mehr in Richtung Flügelinneres verschoben wird, wenn sie einmal außen platziert wurde, da eine Verschiebung

Iterationen	Schwärme	Partikel	Restriktionserfüllung	Gesamtkabellänge [m]
1000	1	100	nein	-
5000	1	100	nein	-
10000	1	100	nein	-
1000	1	300	nein	-
5000	1	300	nein	-
10000	1	300	nein	-
1000	1	500	nein	-
2000	1	500	nein	-
1000	1	1000	ja	763

Tabelle 5.1: Untersuchungsreihe für die Positionierung von Tragflügelkomponenten.

in die korrekte Richtung zu einer plötzlichen Kollision mit der Flügelhaut führt, was eine schlechtere Gesamtbewertung des Zustandes zur Folge hat. Für die abschließende Auswertung wurde die Kollision mit der Flügelhaut daher vernachlässigt. Dies bot weiterhin den Vorteil einer schnelleren Berechnung in jeder Iteration.

Mit der durchgeführten Untersuchungsreihe kann gezeigt werden, dass es prinzipiell möglich ist, Packingaufgaben mithilfe der Partikel-Multi-Schwarm-Optimierung zu lösen, auch wenn die Effizienzfrage offen bleibt. Hohe Partikelanzahlen tragen dabei mehr zum Erfolg der Optimierung bei als hohe Iterationswerte, was einen ersten Einblick gibt. Die genauen Parametereinstellungen für eine leistungsfähige und effiziente Suche müssen allerdings detaillierter auf diesen Anwendungsfall angepasst werden und bedürfen einer ausgiebigeren Untersuchung. Die in der vorgestellten Versuchsreihe aufgefundene valide Lösung ist in Abb. 5.24 dargestellt.

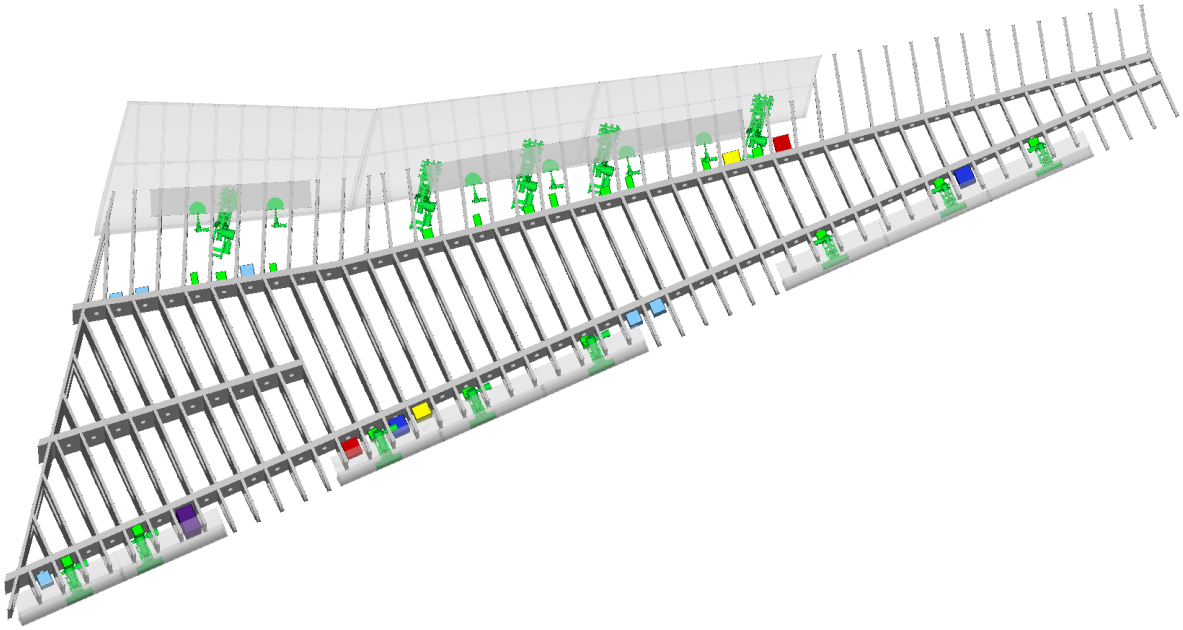


Abbildung 5.24: Gesamtergebnis der Partikel-Multi-Schwarm-Optimierung.

5.3 Packing und Piping auf einem Satellitenantriebspaneel

Bei der Integration von komplexen Systemen innerhalb des Systementwurfs hat sich eine intuitive Reihenfolge von Aufgaben etabliert, welche die aufeinander folgenden Schritte *Packing - Piping - Routing* beinhaltet. Diese Reihenfolge ergibt sich zum einen aus der gegenseitigen Abhängigkeit der einzelnen Schritte, zum anderen ist sie aber auch auf die im Ingenieurentwurf notwendige Pragmatik zurückzuführen, in endlicher - und meist knapp bemessener - Zeit eine ausreichend gute (aber dennoch suboptimale) Lösung zu finden. Die Folge daraus ist, dass die einzelnen Teilaufgaben also separat ausgeführt und nur an definierten Übergabe-Schnittstellen untereinander vernetzt werden. Zwar erfolgen Rückkopplungen zwischen den einzelnen Schritten, diese finden jedoch meist nur im Falle der zwingenden Notwendigkeit statt. Im vorhergehenden Abschnitt konnte gezeigt werden, dass die Abarbeitung dieser nachfolgenden Teilaufgaben durchaus automatisiert werden kann. Dabei ist aber ebenso klar geworden, welche schier unendliche Vielfalt durch reine Iteration abgedeckt werden muss. Der vorliegende Abschnitt befasst sich mit einer Methode, in der klassischerweise erst zu einem späteren Zeitpunkt betrachtete Aspekte bereits frühzeitig einfließen können, um aus lokal vielversprechenden Bereichen des Entwurfsraums auszubrechen und globalere Gebiete aufzufinden. Konkret soll ein Packingprozess vorgestellt werden, der neben der Konfiguration, also Positionierung und Ausrichtung von Komponenten parallel auch deren Verrohrung berücksichtigt. Dieser Prozess soll anhand der Auslegung eines Satellitenantriebspaneels vorgestellt werden. Das Satellitenantriebspaneel bietet sich hierbei besonders als Beispiel an, da bereits Entwurfssprachen zu dessen Entwurfsautomatisierung existieren. Durch die parallele Packing-Piping-Auslegung kann der noch nicht vollständig digitalisierte und automatisierte Entwurfsprozess von Satellitenantriebspaneelen weiter ausgebaut werden.

Der für die vorgestellte Entwurfssprache notwendige Input umfasst die logischen Systemarchitekturen von Satellitenantrieben, welche üblicherweise in Form von sog. *Flow Schematics* dargestellt werden, sowie deren zugehörige Komponentengeometrien. Alle genannten erforderlichen Informationen können aus dem zuvor in den ausgeführten Entwurfssprachen erzeugten und erweiterten Entwurfsgraphen ausgelesen werden. Für die entwickelte Entwurfssprache zur Positionierung von Satellitenantriebskomponenten unter Berücksichtigung ihrer Verrohrung wird zunächst ein zweidimensionales Ersatzproblem eingeführt. In diesem sollen alle Problementitäten möglichst einfach repräsentiert werden, daher werden die Komponenten durch Rechtecke und alle Verrohrungen durch Linien repräsentiert. Zur Berechnung der Rechtecke kann die in Abschnitt 4.2 vorgestellte Entwurfssprache verwendet werden. In Abbildung 5.25 ist für eine Auswahl an Bauteilen deren zweidimensionales Modell dargestellt, das automatisiert abgeleitet wurde. Weiterhin werden die Eingänge und Ausgänge der einzelnen Komponenten zuvor manuell ausgelesen und in einer Bibliothek abgespeichert.

Im Folgenden soll die Modellbildung für den parallelen Packing- und Verrohrungsansatz näher betrachtet werden. Die Freiheitsgrade sind für jede Antriebskomponente und jedes Rohr eine kontinuierliche Verschiebung entlang der globalen x-Koordinate und die kontinuierliche Verschiebung entlang der globalen y-Koordiante. Der rotatorische Freiheitsgrad wird durch vier boolesche Zustände für die Winkel 0°, 90°, 180° und 270° repräsentiert, von denen genau einer eintritt. Auf diese Weise wird im Modell abgebildet, dass Komponenten eines Satellitenantriebs möglichst rechtwinklig positioniert werden, um rechtwinklige Biegungen in der Verrohrung zu erhalten, was die Verwendung von standardisierten Rohrbauteilen erlaubt. Im Modell gehen folglich pro Komponente vier vorab um die jeweiligen Rotationswerte gedrehte Geometrien ein. Die automatisierte Auswahl einer der rotierten Varianten während der Optimierung entspricht der Auswahl eines Rotationswinkels. Zur parallelen Auslegung der Verrohrung müssen außer-

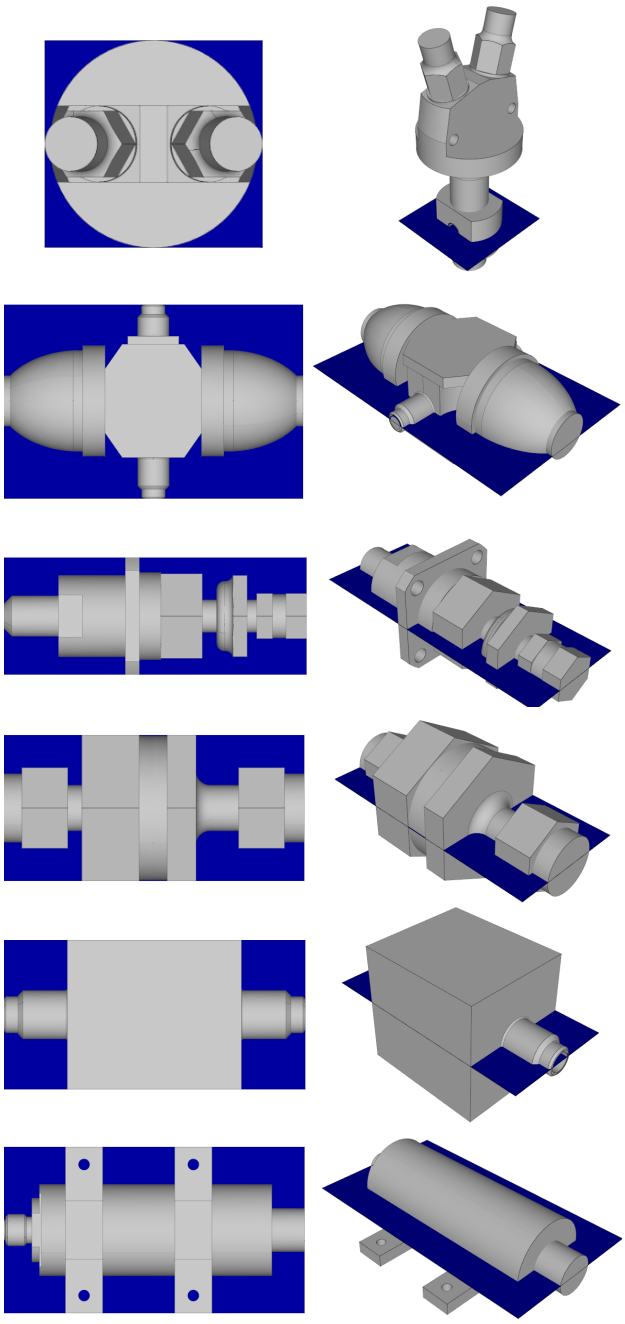


Abbildung 5.25: Repräsentation der Satellitenantriebskomponenten.

dem die Rohrlängen variabel sein. Für jedes Rohr geht folglich ein Skalierungsfreiheitsgrad in das Modell mit ein, welcher unabhängig von einer Raumrichtung die Rohrlänge skaliert. Die Repräsentation der Rohrgeometrie als Linie erlaubt, dass keine Unterscheidung der Skalierungsrichtung in x und y vorgenommen werden muss, da einer der Werte automatisch zu null wird. Darüber hinaus müssen die Freiheitsgrade des Paneels berücksichtigt werden. Das Satellitenpaneel wird als Rechteck mit variabler Breite und Länge repräsentiert. Dafür wird ein Rechteck mit initialen Einheitslängen modelliert, dessen Maße über die Skalierungsfreiheitsgrade in x-Richtung und y-Richtung ausgedehnt werden können. Zuletzt muss sichergestellt werden, dass die Schnittstellen des Antriebssystems über die Systemgrenze hinaus auf den Außenrändern des Paneels positioniert werden. Die Eingänge aus dem Tank und die Ausgänge zu den Antrieben werden als Punkte dargestellt, welche auf den Kanten der Paneelrechtecks positioniert werden. Auch hier wird die Anschlussrichtung der zugehörigen Rohre automatisiert an die jeweilige Paneelseite angepasst.

Sind Bauteile adjazent und folgen damit direkt aufeinander, werden zwischen deren Eingangs- und Ausgangspunkten Rohre eingeplant. Die Nebenbedingungen des Gesamtmodells stellen dann sicher, dass die jeweiligen Punktpaare aufeinander liegen. Während bei Varianten mit einem Eingang und einem Ausgang genau spezifiziert werden kann, welche Komponenten- und Rohrpunkte zusammengehören, existieren auch Bauteile mit mehr als zwei Schnittstellen, bei denen keine eindeutige Zuordnung existiert. Dies wird in der vorgestellten Entwurfssprache berücksichtigt, d.h. die Zuordnung findet in diesen Fällen innerhalb der Optimierung automatisiert statt. Das entwickelte Modell berücksichtigt für die Verrohrung Rohre mit maximal zwei Biegungen. Hierfür können bis zu drei gerade Rohrstrecken zwischen jedem Bauteilpaar im Modell verwendet werden, wobei die durch den/die Anwender/-in vorgegebene Anzahl der Strecken dem Maximalwert entspricht. Die tatsächliche Anzahl an Biegungen wird automatisiert während der Optimierung berechnet. Die vorherige Einschränkung der zu berücksichtigenden Streckenanzahl ist vor allem bei begrenzt zur Verfügung stehenden Ressourcen von Vorteil, denn sie ermöglicht eine Anpassung an die Problemkomplexität, sodass bei einer hohen Anzahl an Bauteilen zur Laufzeitersparnis auch einfachere Rohrführungen ohne Biegungen ver-

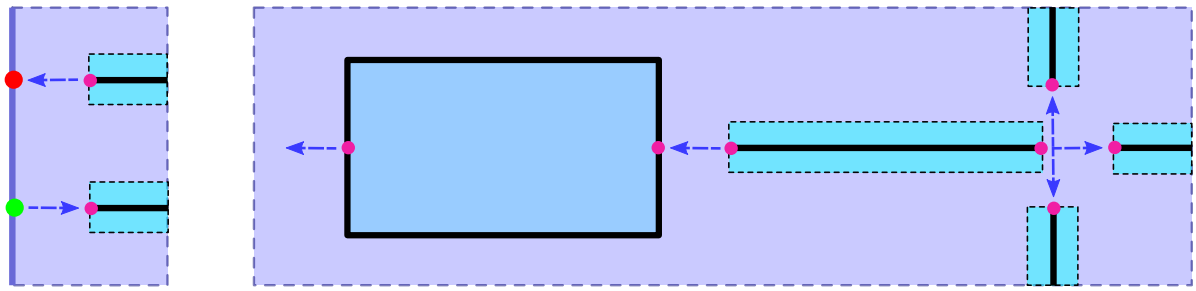


Abbildung 5.26: Nebenbedingungen für das Paneelmodell. Links: Schnittstellen zum Paneelsystem und Ein- und Austrittsrichtungen jeweils in grün und rot. Rechts: Komponenten und Rohre sowie erlaubte Richtungen an deren Schnittstellen.

wendet werden können. Bei der Verwendung von mehr als einem Rohrabschnitt muss allerdings sichergestellt werden, dass der Biegeradius realisierbar ist. Hierfür müssen die Mindestlängen der Rohre dem minimalen doppelten Biegeradius entsprechen. Weiterhin berücksichtigen die Randbedingungen des Modells, dass Rohrabschnitte ausschließlich geradeaus in eine Komponente führen dürfen. Das bedeutet, dass die Rotationen der ein- und ausgehenden Rohre und er zugehörigen Bauteile automatisiert aufeinander abgestimmt werden. In Abb. 5.26 sind die erlaubten Ein- und Austrittsrichtungen zwischen Bauteilen und Rohren visualisiert.

Weitere Anforderungen an das Modell sind, dass die Antriebskomponenten und deren Verrohrung keinen gegenseitigen Kollisionen unterliegen, d.h. dass deren Geometrien verschneidungsfrei positioniert werden. Da die Rohre infolge der vereinfachten Liniendarstellung keine Ausdehnung in radialer Richtung haben, wird ein Minimalabstand zu allen potenziellen Kollisionspartnern definiert. Dieser muss mindestens dem Rohrradius entsprechen. In Abb. 5.26 sind die theoretischen Maße der Rohre visualisiert. Als Kollisionspaare gelten alle Antriebskomponenten untereinander, alle Rohrstrecken untereinander sowie die Rohrstrecken und Antriebsbauteile, welche nicht zusammenhängen. Sind Rohre und Bauteile gekoppelt, stellen die gegenseitige Ausrichtung und die Gleichheit der Kontaktpunkte bereits eine Kollisionsfreiheit sicher, wodurch keine zusätzlichen Restriktionen definiert werden müssen. Außerdem müssen alle Bauteile und Rohre auf dem Satellitenpaneel Platz finden. Diese Forderung kann unter Verwendung des Packingframeworks einfach durch Einsatz der Inklusionsbedingungen repräsentiert werden. Die Paneelmaße passen sich dann innerhalb der Optimierung automatisiert an die Bauteile und Rohre an.

Auch bauteilspezifische Randbedingungen werden durch das Modell abgebildet. Beispielsweise fungieren neben den Systemeingängen und Ausgängen weitere Antriebskomponenten als Schnittstellen nach außen. Die Ventile zum Befüllen und Entleeren müssen demnach ebenso am Rand des Satellitenpaneels positioniert werden. Eine weitere Anforderung ist bei der Verwendung von pyrotechnischen Ventilen gegeben. Diese müssen immer so mit T-Rohren verbunden werden, dass die bei einer Sprengung möglicherweise abfallenden Splitter von der Leitungswand der T-Rohre abgefangen werden und nicht in das System eindringen. Über die grundlegenden Modellrestriktionen hinaus sind weitere optionale Nebenbedingungen integriert, welche durch den/die Anwender/-in zugeschaltet werden können. Beispielsweise ist es möglich, Mindestabstände zu definieren, welche zwischen Bauteilen vorherrschen müssen. Dafür kann ein Minimalabstand für die Kollisionsfreiheit verwendet werden, genauso wie er auch für die Ausdehnung der Rohre in radialer Richtung zum Einsatz kommt. Eine weitere optionale Forderung ist die Positionierung aller Ausgänge oder Eingänge auf derselben Seite des Paneels. Auch diese kann wahlweise aktiviert werden.

Die Zielfunktion der Entwurfssprache liegt in quadratischer Form vor und dient der Minimierung der Satellitenpaneelfläche. Diese wird von innen heraus durch die einzelnen Bauteile beschränkt und kann von außen durch die Vorgabe von maximalen Ausdehnungen begrenzt werden. Daneben können auch die Rohrlängen in der Zielfunktion berücksichtigt werden, welche einen linearen Anteil hinzufügen. Die Gewichtungsfaktoren werden von dem/der Anwender/-in übergeben. Alle weiteren Nebenbedingungen liegen in linearer Form vor, weshalb eine mathematische Optimierung mithilfe einer externen Optimierungssoftware zum Einsatz kommt. Für die im Folgenden vorgestellten Berechnung berücksichtigt die Zielfunktion die Paneelfläche mit einer Gewichtung von 0.9 und die Gesamtrohrstrecke mit einer Gewichtung von 0.1. Die Untersuchungsreihe umfasst das Packing eines Kaltgassystems und eines Monergolsyszems, jeweils zunächst unter Berücksichtigung einer biegefreien Verrohrung und nachfolgen maximal einer zulässigen Biegung pro Verrohrungsstrecke zwischen zwei Bauteilen.

In Abb. 5.27 ist das Ergebnis der Berechnung eines Kaltgassystems mit einer biegefreien Verrohrung dargestellt. Die Berechnungszeit beträgt hierbei 30 Sekunden bis das nachgewiesen global optimale Ergebnis mit einer Paneelfläche von rund 548 cm^2 für die Aufgabestellung ermittelt werden kann. Aufgefunden wird das Ergebnis bereits nach 23 Sekunden, der Nachweis der globalen Optimalität benötigt eine weitere Berechnungszeit von 7 Sekunden. Insgesamt werden 9 Lösungen evaluiert, die erste unter den gegebenen Nebenbedingungen valide suboptimale Lösung wird bereits nach 19 Sekunden erzielt. Abb. 5.28 stellt das Ergebnis der erweiterten Berechnung unter Berücksichtigung von maximal einer erlaubten Biegung pro Rohrstrecke dar. Die erste unter den Randbedingungen gegebene valide Lösung wird nach 122 Sekunden errechnet, die gesamte Berechnung wird schließlich nach 7200 Sekunden (2 Stunden) abgebrochen. Zu diesem Zeitpunkt kann das dargestellte Ergebnis gewonnen werden, welches lediglich für 49.3 % des gesamten Suchraums als global nachgewiesen werden kann. Tatsächlich wird diese Lösung bereits nach 533 Sekunden errechnet und bleibt bis zur Erreichung des Zeitlimits die bis dato beste aufgefundene Lösung. Insgesamt können innerhalb der gegebenen Rechenzeit 36 Lösungen evaluiert werden. Mit einer Fläche von rund 379 cm^2 kann trotz der Rechenzeitlimitierung eine deutliche Verbesserung gegenüber der biegefreien Variante erzielt werden. Zu Testzwecken wird dieselbe Berechnung ein weiteres Mal ausgeführt. Das Ergebnis ist Abb. 5.29 zu entnehmen. Obwohl mit 31 Lösungen insgesamt weniger Varianten getestet werden, kann die erste Lösung bereits nach 120 Sekunden aufgefunden werden, während nach 1665 Sekunden sogar ein etwas besseres Ergebnis mit rund 372 cm^2 errechnet werden kann, das nach Ablauf der gegebenen Rechenzeit für 50.1 % des Lösungsraums als optimal nachgewiesen wird. Es kann folglich beobachtet werden, dass exakt gleiche Eingangsbedingungen durchaus zu unterschiedlichen Ergebnissen führen können. Hierin äußert sich der Nichtdeterminismus bei Einsatz von externer Optimierungssoftware, wie in Abschnitt 4.3.1 bereits beschrieben.

Ein weiterer Testlauf enthält die Optimierung mit maximal zwei erlaubten Biegungen pro Rohrstrecke. Die Berechnung wird aufgrund der deutlich erhöhten Anzahl von (Hilfs-) Variablen und Nebenbedingungen auf 48 Stunden begrenzt. In dieser Zeit werden 31 Lösungen evaluiert. Trotz des hohen Zeitlimits kann nach Abbruch des Optimierungslaufs keine Verbesserung gegenüber den vorhergehenden Lösungen erzielt werden. Auch die theoretisch erzielbare Lösung aus einer der vorhergehenden Berechnungen mit maximal zwei Biegungen pro Verrohrungsstrecke kann nicht aufgefunden werden, was auf den polynomiellen Anstieg an Variablen und Gleichungen durch die gegenseitige Kollisionsprüfung aller Komponenten (also der Bauteile und der einzelnen geraden Rohrstücke) untereinander sowie den nichtlinearen Anstieg der Rechenzeit im Verhältnis zu der Variablen- und Gleichungsanzahl zurückzuführen sein sollte. Bei der gegebenen Rechenleistung der vorliegenden Hardware stößt die mathematische Optimierung für das Packing von Satellitenantriebskomponenten hier auf eine erste Grenze.

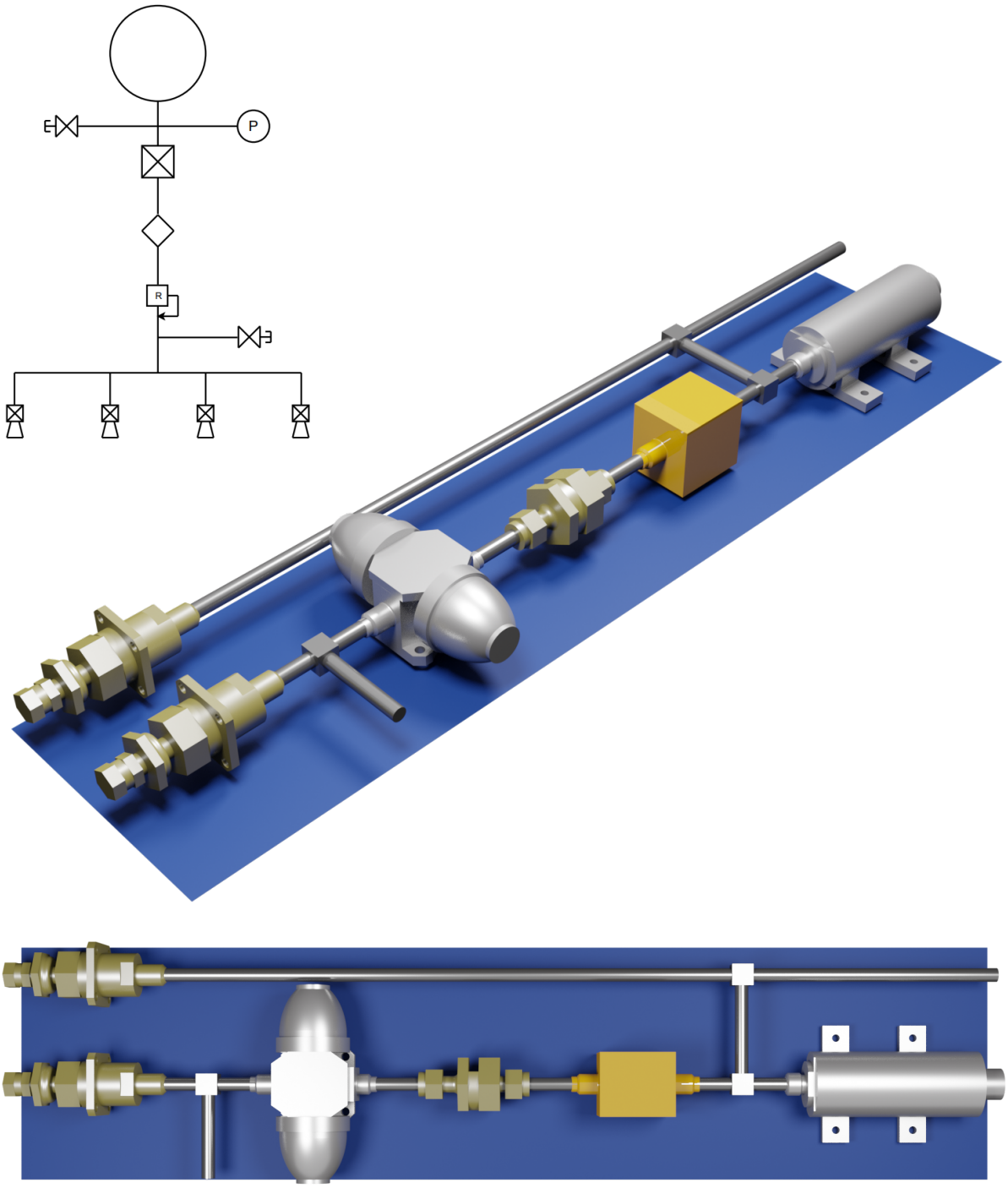


Abbildung 5.27: Lösung für ein Kaltgassystem mit einer biegefreien Verrohrung. Erzielte Paneelfläche: etwa 548 cm^2.

In Abb. 5.30 ist das Ergebnis der Berechnung eines Monergolsystems mit einer biegefreien Verrohrung dargestellt. Die erste suboptimale Lösung wird nach 38 Sekunden aufgefunden und das Ergebnis mit einer Paneelfläche von ca. 457 cm^2 kann nach einer Berechnungszeit von 277 Sekunden als global optimal nachgewiesen werden. Aufgefunden wird das Optimum bereits nach 42 Sekunden, die restliche Rechenzeit wird für den Nachweis der globalen Optimalität aufgewendet. Insgesamt müssen lediglich 2 Lösungsvarianten evaluiert werden.

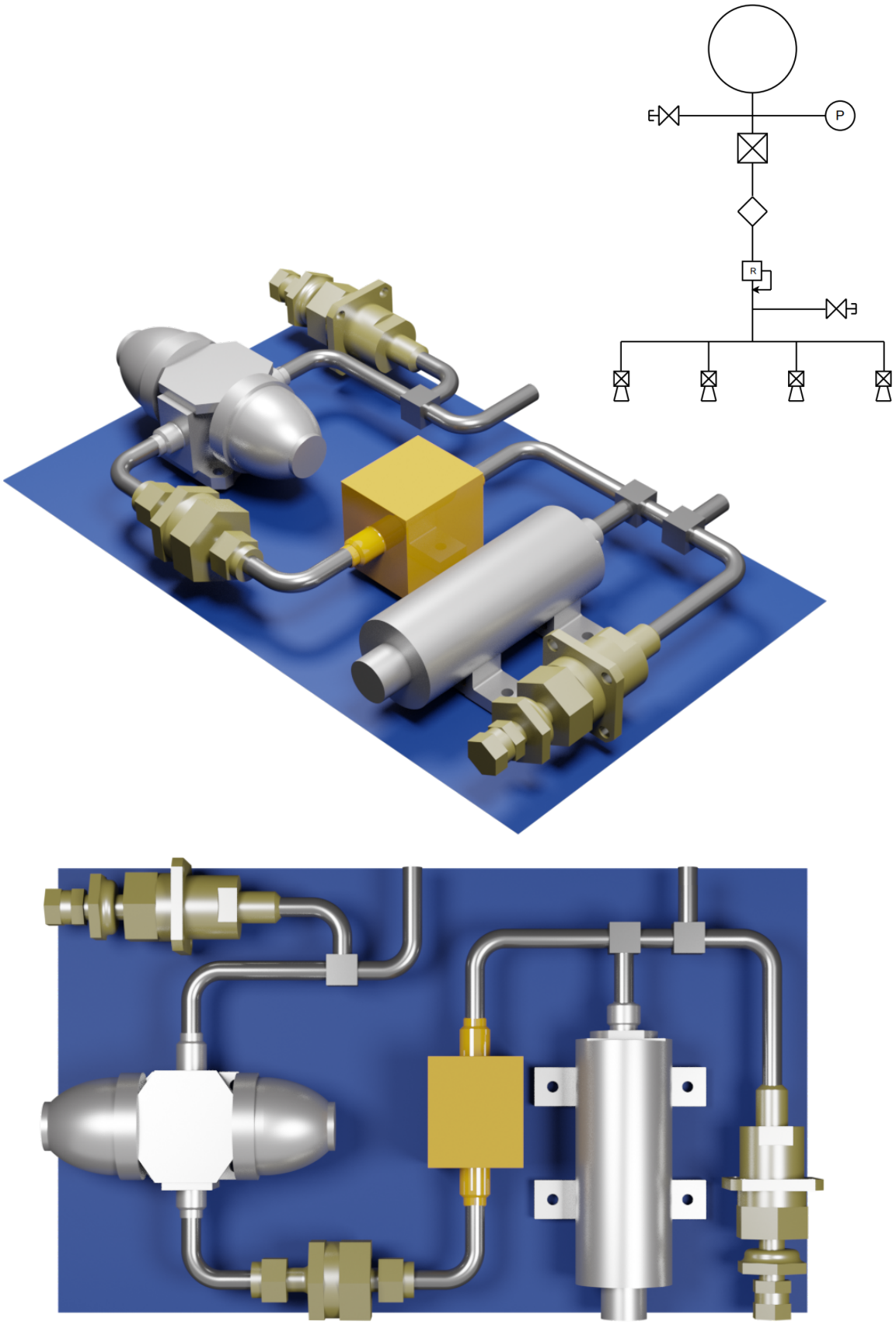


Abbildung 5.28: Lösung für ein Kaltgassystem mit einer zulässigen Biegung pro Verrohrungsstrecke. Erzielte Paneelfläche: etwa 379 cm^2.

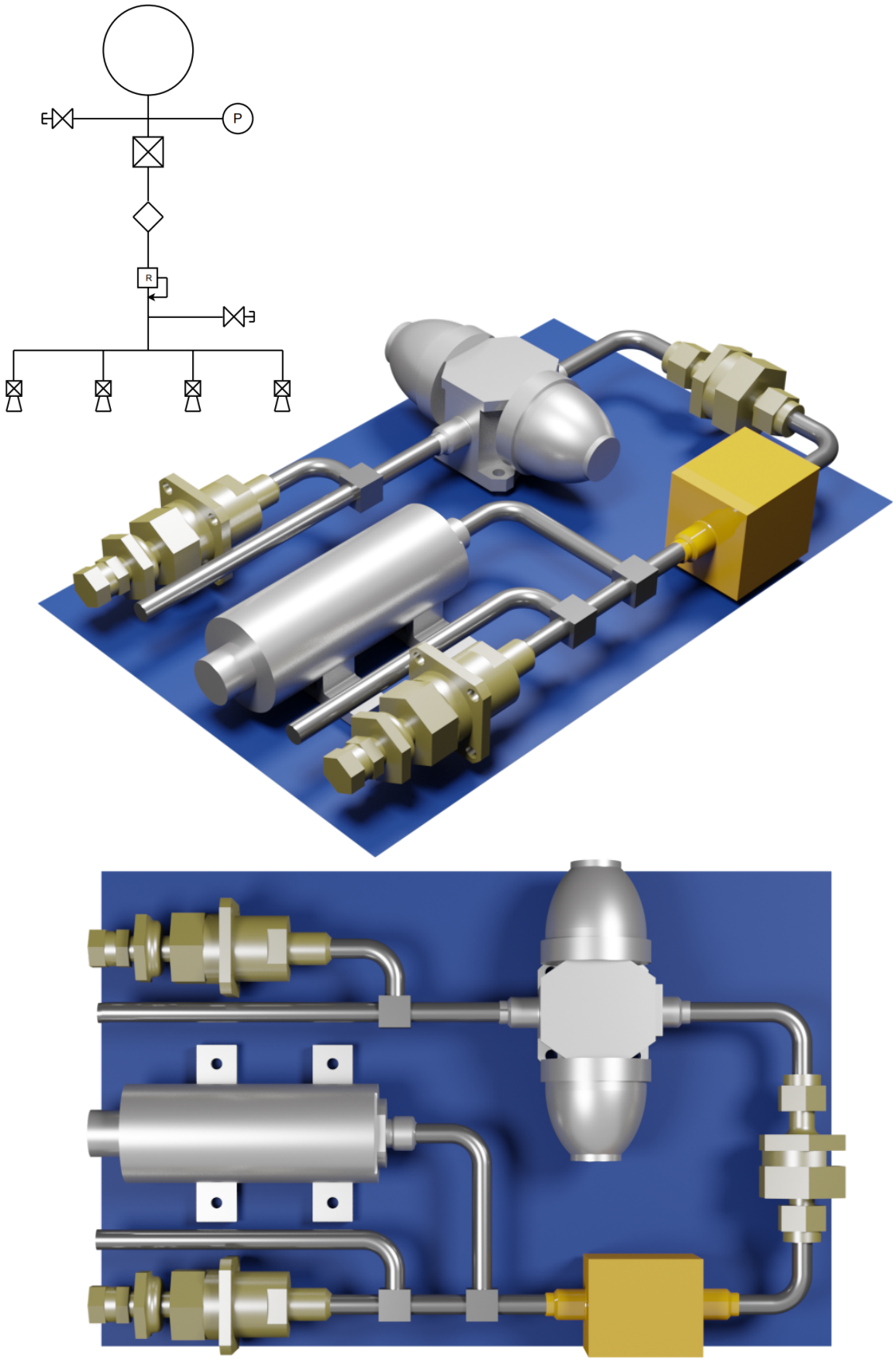


Abbildung 5.29: Weitere Lösung für ein Kaltgassystem mit einer zulässigen Biegung pro Verrohrungsstrecke. Erzielte Paneelfläche: etwa 372 cm^2.

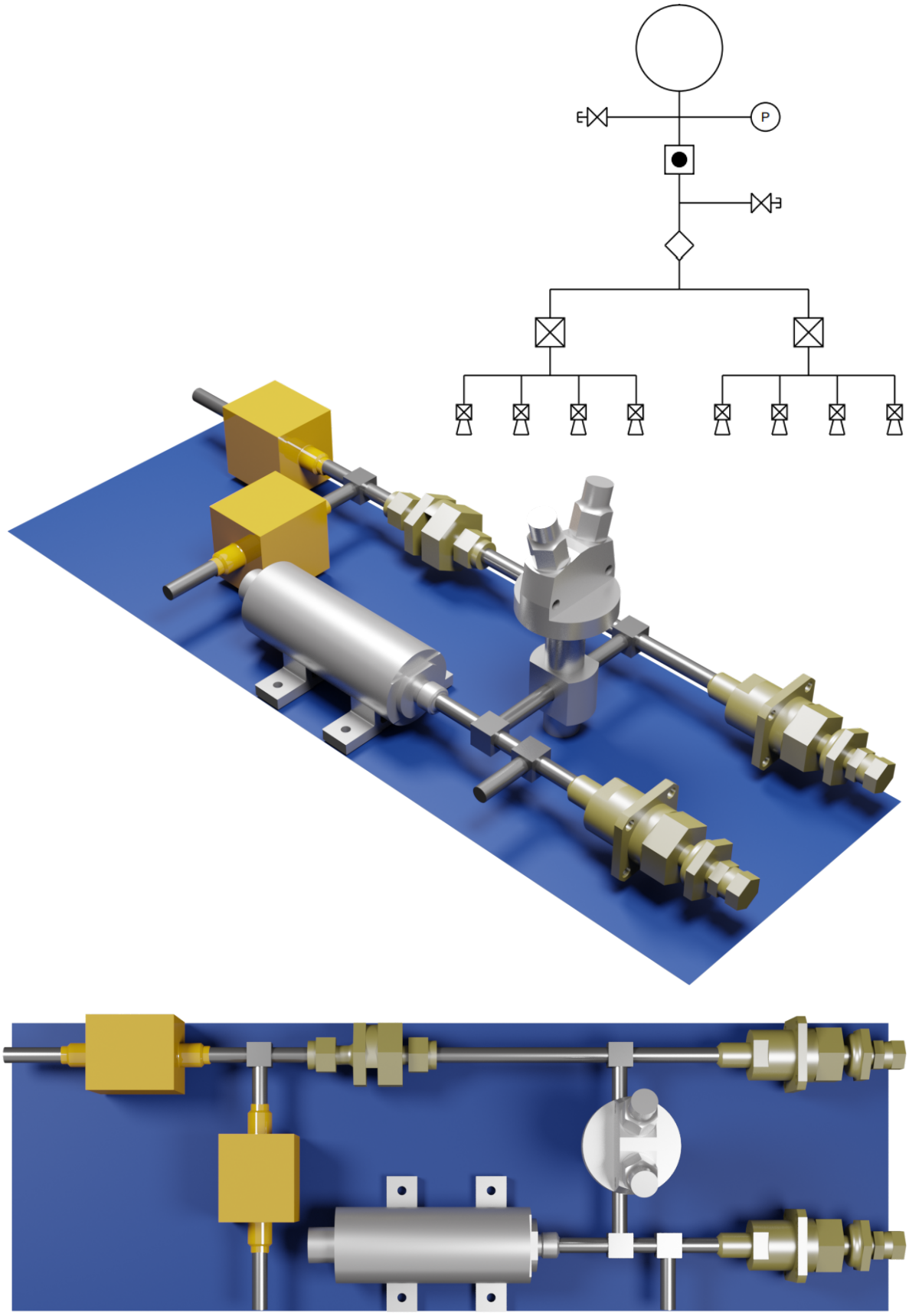


Abbildung 5.30: Lösung für ein Monergolsystem mit einer biegefreien Verrohrung. Erzielte Paneelfläche: etwa 457 cm^2.

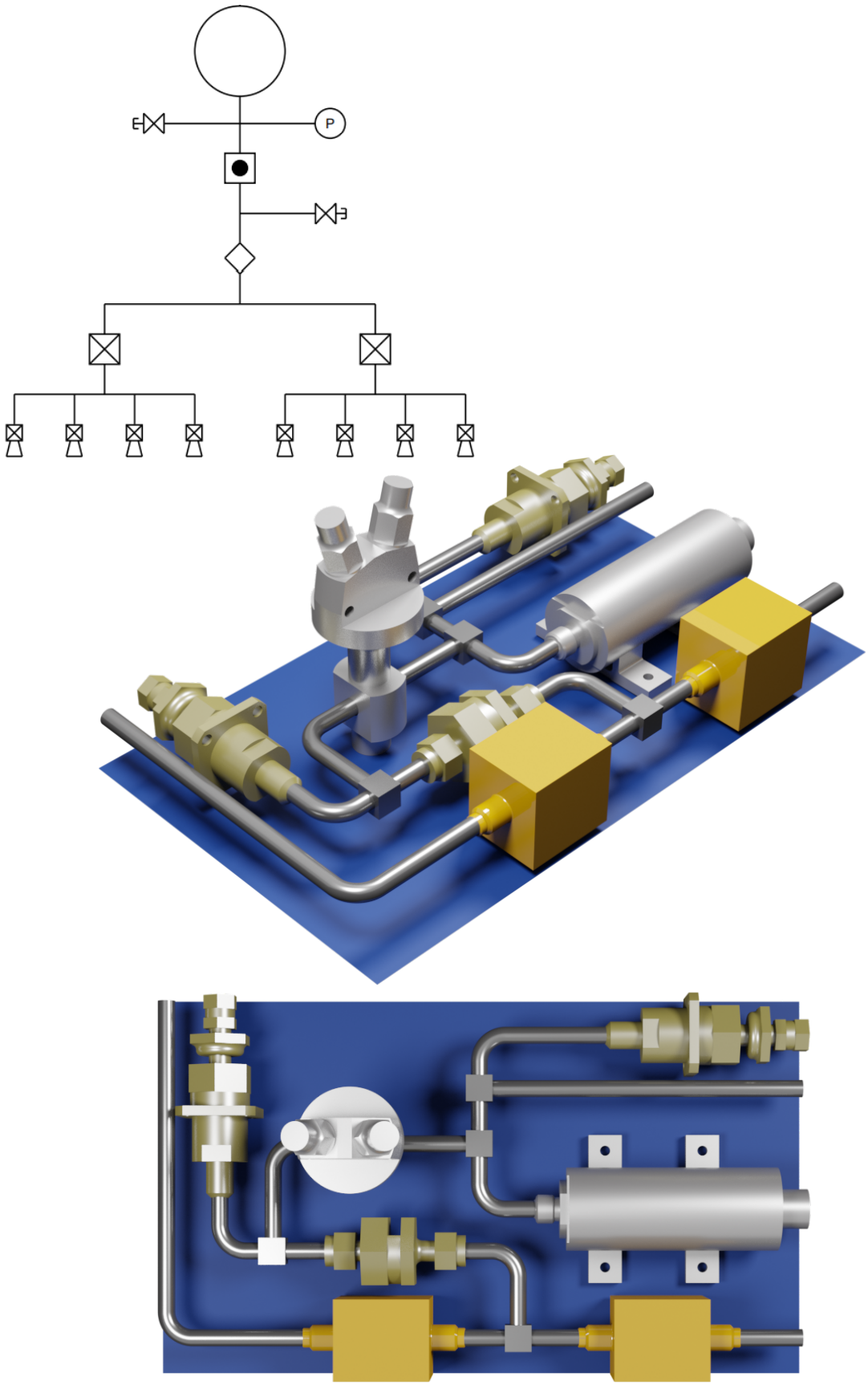


Abbildung 5.31: Lösung für ein Monergolsystem mit einer zulässigen Biegung pro Verrohrungsstrecke. Erzielte Paneelfläche: etwa 361 cm^2.

Das Packing des Monergolsystems mit maximal einer erlaubten Biegung pro Rohrstrecke zwischen zwei Bauteilen erfordert zum Auffinden einer suboptimalen Lösung eine deutlich höhere Rechenzeit. Während eine erste suboptimale Lösung nach 1576 Sekunden aufgefunden wird, kann das beste gefundene Ergebnis nach 85 Minuten Rechenzeit ausgelesen werden. Dieses weist eine Paneelfläche von rund 361 cm^2 auf, was eine Verbesserung gegenüber der biegefreien Variante darstellt. Das Suboptimum kann allerdings lediglich für 20.6 % des gesamten Suchraums als global nachgewiesen werden, was auch nach 51 aufgefundenen Lösungsvarianten und einer weiteren Laufzeit von 12 Stunden keiner Änderung unterliegt. Die Ergebnisse hierzu sind Abb. 5.31 zu entnehmen. Auch für das Monergolsystem wird ein weiterer Testlauf mit maximal zwei erlaubten Biegungen pro Rohrstrecke durchgeführt. Wie bei der Auslegung des Kaltgassystems können auch hierbei auf der zur Verfügung stehenden Hardware keine Ergebnisse in adäquater Rechenzeit erzielt werden. Nach einem vordefinierten Zeitlimit von 24 Stunden kann keine Verbesserung gegenüber der Optimierungsvariante mit maximal einer Biegung pro Verrohrungsstrecke erzielt werden.

Als weiteres Beispiel wird weiterhin das Packing für ein einfaches Diergolsystem durchgeführt. Aufgrund der aus den vorhergehenden Packingoptimierungen gewonnenen Erkenntnisse wird für die vorliegende Anzahl an Komponenten eine Vereinfachung eingeführt. Dabei wird die Eigenschaft des Diergolssystems ausgenutzt, dass dieses aus drei separate Teilbereichen besteht, welche durch außerhalb des Paneels liegende Tanks getrennt sind. Für die nachfolgende Packingoptimierung wird nun für jeden Teilbereich eine Zone eingeführt, welche die Bauteile des jeweiligen Bereiches enthält. Die Zonen müssen innerhalb des Satellitenpaneels unter dessen Flächenminimierung positioniert und dimensioniert werden. Die Anzahl der Inklusionsbedingungen wird dadurch nur marginal erhöht, da die Nebenbedingungen für das Enthaltensein der Komponenten innerhalb der Konturen des Paneels durch das Enthaltensein innerhalb der zugehörigen Zonen ersetzt werden und lediglich die Gleichungen für das Enthaltensein der Zonen innerhalb der Paneelgrenzen das Modell erweitern. Weiterhin wird die Kollisionsfreiheit der Zonen untereinander eingeführt. Im Gegenzug können die Nebenbedingungen für die Kollisionsvermeidung der Komponenten untereinander erheblich reduziert werden, da ausschließlich Komponenten innerhalb einer Zone miteinander getestet werden müssen. Mithilfe der eingeführten Vereinfachung kann innerhalb von 2520 Sekunden eine erste valide suboptimale Lösungen gefunden werden. Nach 36 Stunden sind 38 Lösungsvarianten evaluiert und es liegt eine Lösung vor, welche für 20.1 % des gesamten Lösungsraumes als optimal nachgewiesen ist. Diese ist Abb. 5.32 zu entnehmen. Das Packing der Diergolsystemkomponenten unter Berücksichtigung von maximal einer Biegung pro Verrohrungsstrecke erzielt nach 36 Stunden keine Lösung und erfordert höhere Rechenzeiten, weitere Vereinfachungen oder eine bessere Rechenleistung.

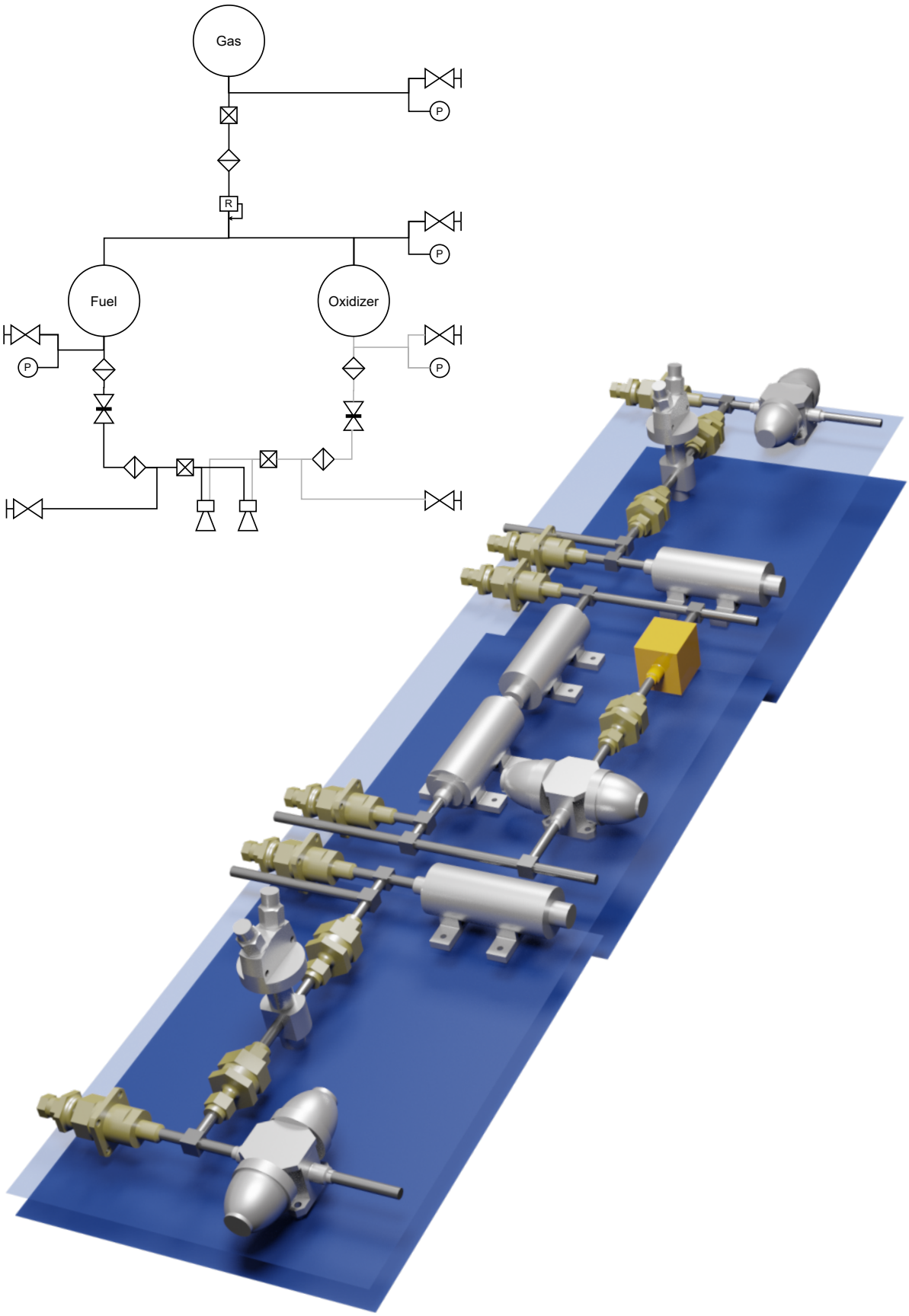


Abbildung 5.32: Lösung für ein Diergolsystem mit einer biegefreien Verrohrung. Erzielte Paneelfläche: etwa 2010 cm^2. Die einzelnen Paneelzonen sind treppenartig visualisiert.

KAPITEL 6

Zusammenfassung & Fazit

Die Erhaltung der wirtschaftlichen Wettbewerbsfähigkeit bei gleichzeitigem umweltorientierten Handeln stellt Unternehmen vor die Herausforderung, immer komplexere Systeme verwirklichen zu müssen. Die im Produktentwurf vorherrschende Praxis, steigende Systemkomplexität durch Aufgabenteilung zu beherrschen, stößt in Systemen von heutiger Größenordnung allerdings auf Grenzen. Diese sind vor allem auf die der Aufgabenteilung inhärenten Defizite zurückzuführen und umfassen aufgrund von komplementärem Hintergrund und Fachwissen sowie unterschiedlicher Zielvorstellungen und Entwicklungszeiten der Entwicklungsteams inkompatible Modelle und Datenformaten, inkohärent weiterentwickelte Modellstände und die inkonsistente Datenhaltung. All diese Missstände führen zu Ressourcenverschwendung, Ineffizienz und Wettbewerbsnachteilen und können in vollem Umfang ausschließlich durch die vollständige Digitalisierung des Produktentwurfes gelöst werden. Eine vollständige Digitalisierung hat den Vorteil einer konsistenten Datenbasis und ermöglicht darüber hinaus die Automation des Entwurfsprozesses, in dem Modellstände des Produktes unter Berücksichtigung digital definierter Anforderungen und Randbedingungen generiert werden können.

Die vorliegende Arbeit befasst sich in diesem Kontext mit der Integrationsphase des Produktentwurfes. Im Mittelpunkt der Aufgabenstellung steht dabei die Entwicklung eines digitalen Frameworks, um die automatisierte Lösung von Layout- und Packingproblemen in vernetzten Systemen zu unterstützen und eine effiziente Generierung und Bewertung von Entwurfslösungen sowie die Erforschung neuer Konfigurationen und deren Optimierung zu ermöglichen. Zur Umsetzung des Frameworks dienen die graphenbasierten Entwurfssprachen, welche die Möglichkeit bieten, den entwickelten digitalen Integrationsprozess in eine Vielzahl bereits bestehender digitaler Entwurfsschritte in Form von Entwurfssprachen einzubetten.

In Abschnitt 2.2 der vorliegenden Arbeit wurden zunächst die sechs Hauptanforderungen an das zu entwickelnde Framework abgeleitet, welche die zentralen Grundsteine für dessen nachfolgende Entwicklung legen. Diese sind die Möglichkeit zur konsistenten Wissensrepräsentation, eine ausreichend hohe Flexibilität, die Erweiterbarkeit und Anpassungsfähigkeit des Frameworks sowie dessen Modularität, die Integrationsfähigkeit der im Framework zur Verfügung gestellten Methoden in den Gesamtentwurf sowie die Domänenunabhängigkeit bei der Anwendung und die Visualisierungsfähigkeit der Ergebnisse für eine einfache und dem/der Anwender/-in zugängliche Auswertung.

Darauf aufbauend ist es in Kapitel 4 gelungen, einen Ansatz für ein Framework vorzustellen, welches diesen Anforderungen genügt. Den Kern des entwickelten Frameworks bildet die

Entwurfssprache für Packingprobleme, welche alle den Layout- und Packingprozessen innewohnende Elemente umfasst und die digitale Modellierung und ganzheitliche Darstellung von packingspezifischen Problemen durch den/die Anwender/-in ermöglicht. Diese ruht auf zwei weiteren Säulen. Zum einen wurde für die Lösung der Layout- oder Packingprobleme im Sinne einer Optimierungsaufgabe die *Entwurfssprache zur Optimierung* entwickelt. Diese wurde aus der Notwendigkeit für ein Packingframework heraus geboren, konnte jedoch als unabhängiges Modul umgesetzt werden, sodass sie auch bei anderweitigen Optimierungsproblemen Anwendung finden kann. Zum anderen wurde als Erweiterung die *Entwurfssprache für Geometrieapproximation* entwickelt, welche das Einlesen von Geometrien aus beliebigen Datenformaten ermöglicht und eine Reihe an geometrischen Vereinfachungen bietet, um eine geeignete und problemangepasste Repräsentation der Packinggeometrien zu erhalten. Alle drei Säulen des Packingframeworks sind in der vorliegenden Arbeit als eigenständige Ontologien modelliert worden, welche das Repertoire der bestehenden Basisontologien in graphenbasierten Entwurfssprachen erweitern. Der erfolgreiche Einsatz des entwickelten Frameworks konnte in Kapitel 5 demonstriert werden und soll im Kontext der abgeleiteten Anforderungen im Folgenden genauer diskutiert werden.

6.1 Diskussion der Ergebnisse

Die in der vorliegenden Arbeit erarbeiteten Anforderungen und Randbedingungen an ein fundamentales, ganzheitliches und systemorientiertes Verfahren zur Lösung von Layout- und Packingproblemen bilden den Kern für dessen erfolgreiche Umsetzung. Das entwickelte und implementierte Framework erfüllt diese Forderungen im Grundsatz, was anhand der Anwendungsbeispiele demonstriert werden konnte. Im Folgenden sind die Erkenntnisse, welche aus den Anwendungsbeispielen hinsichtlich der Umsetzung der Forderungen gewonnen werden konnten, im Einzelnen dargestellt.

Konsistente Wissensrepräsentation Eine konsistente Wissensrepräsentation ist dann gegeben, wenn die Modellierung und Verarbeitung von Rand- und Nebenbedingungen aus unterschiedlichsten naturwissenschaftlichen Gebieten möglich ist. Diese Forderung wird durch die Nutzung von graphenbasierten Entwurfssprachen erfüllt. Die vorgestellten Beispiele demonstrieren dies anhand einer Fülle von geometrischen und physikalischen Bedingungen, welche problemlos miteinander kombiniert wurden. In *5.2 Packing in einem Flugzeugtragflügel* beispielsweise konnte das konsistente Zusammenspiel der Aufgabenbereiche Packing, Piping und Routing nachgewiesen werden, welchem eine konsistente Wissensrepräsentation zugrunde liegt. In Hinblick auf einen ganzheitlichen Ansatz kann weiterhin davon ausgegangen werden, dass eine konsistente Repräsentation und gemeinsame Behandlung von kontinuierlichen und diskreten Elementen notwendig ist, da sowohl kontinuierliche (Position und Orientierung, Dimensionierung und Abmaße usw.) als auch diskrete Elemente (Topologievarianten, Anzahl der Komponenten, Verbindungen, Kreuzungen usw.) enthalten sind. Das Framework stellt eine Bibliothek bereit, welche sowohl kontinuierliche als auch diskrete, als auch boolesche Variablen umfasst. In *5.3 Packing und Piping auf einem Satellitenantriebspaneel* beispielsweise konnte die gemeinsame konsistente Verarbeitung aufgezeigt werden, indem topologische Variationen der Rohrelemente durchgeführt wurden, die gleichzeitig mit einer kontinuierlichen Translation behaftet waren. Auch die translatorischen Freiheitsgrade der Satellitenantriebskomponenten wurden kontinuierlich dargestellt, während deren rotatorische Freiheitsgrade als diskrete Variablen in das Problem eingingen.

Erweiterbarkeit und Anpassungsfähigkeit Packingaufgaben in vernetzten Systemen erstrecken sich von der bloßen translatorischen Positionierung, über die zusätzliche rotatorische Orientierung bis hin zu der darüber hinausgehenden Veränderung der Gestalt der Einzelelemente und berücksichtigen fast immer die Kollisionsfreiheit von Elementen bis hin zu einer Reihe an geometrischen Größen wie beispielsweise Abständen und beziehen in seltenen Fällen auch physikalische Kriterien aus den Bereichen Mechanik, Thermodynamik, Hydraulik, Optik und viele mehr mit ein. Eine vollständige Repräsentation aller erforderlicher und möglicherweise noch unbekannter Randbedingungen innerhalb des entwickelten Frameworks kann nicht allumfassend geleistet werden. Daher wird die Erweiterbarkeit der im Framework enthaltenen Modelle gefordert. Die Erweiterbarkeit intendiert dabei eine einfache Anpassungsfähigkeit der mithilfe des Frameworks modellierten Basismodelle an die individuellen Problemstellungen über das Hinzufügen von problemspezifischen Anforderungen. In *5.1.2 Packingframework zur Layoutoptimierung im Detail* wurden neben den im Framework bereits vorhandenen Randbedingungen weitere Anforderungen hinsichtlich der Flächenverhältnisse in Form von zusätzlichen Restriktionen manuell modelliert. In *5.3 Packing und Piping auf einem Satellitenantriebspaneel* dagegen wurden zusätzliche Bedingungen im Modell verwendet, um die Rotationen der ein- und ausgehenden Rohre und der zugehörigen Bauteile automatisiert aufeinander abzustimmen. In beiden Beispielen mussten die zusätzlichen Restriktionen lediglich dem Gesamtproblem hinzugefügt werden, wodurch sich die mühelose Erweiterbarkeit der Problemmodelle offenbart.

Modularität Aufgrund der allgemeinen Problemabhängigkeit von Packingproblemen geht die vorliegende Arbeit davon aus, dass es mit hoher Wahrscheinlichkeit nicht möglich ist, jedes Packingproblem derart zu abstrahieren, dass eine einzige Packingstrategie dieses effektiv lösen kann. Es ist folglich unabdingbar, dass ein Packingframework nicht aus einer einzigen starr implementierten Vorgehensweise besteht, sondern viel mehr die Kombination von möglichen vorhandenen Packingstrategien ermöglicht und damit die Modellierung eines problemabhängigen Packings in allen seinen Facetten und Teilbereichen unterstützt. Diese Forderung konnte durch eine adäquate Modularität des entwickelten Frameworks erreicht werden. Dies beinhaltet zum einen die Aufteilung in drei separate Entwurfssprachen jeweils für die Aufgaben *Geometrievereinfachung*, *Optimierung* und *Packing*, sowie deren weitere Unterteilung in unabhängige Module, welche nach Bedarf zusammengestellt werden können. Neben der zweckdienlichen Komposition der Problemmodelle ermöglicht diese Struktur auch die Verwendung von unterschiedlichen Optimierungsstrategien und Optimierungssoftwares zur Lösung der Layout- bzw. Packingprobleme sowie deren Kombination in den jeweiligen Beispielen. Bei der Auslegung von Tragflügelkonfigurationen in 5.2 beispielsweise wurde für die Verteilung von elektronischen Komponenten ein Verfahren vorgestellt, welches auf die integrierte Partikel-Multi-Schwarm-Optimierung zugreift. In einem anderen Ansatz, in dem dieselbe Aufgabe noch um die Verrohrung und Verkabelung erweitert wurde, wurde statt einer unabhängigen Optimierung des Packings eine Optimierungsstrategie verfolgt, welche mithilfe einer Optimierungssoftware iterativ alle Entwurfssprachen des Gesamtentwurfs aufruft. Eine Kombination der beiden Ansätze wäre mithilfe des Frameworks ohne großen Aufwand möglich. Lediglich aufgrund der hohen Laufzeiten wurde darauf verzichtet.

Flexibilität Bei der automatisierten Lösung von Packingproblemen ist zu berücksichtigen, dass die gegebene Hardware eine entscheidende Rolle in der Rechenleistung spielt und gegebenenfalls das Packingmodell an diese angepasst werden muss. Gerade im Falle von System-of-Systems Problemen ist es daher häufig notwendig schrittweise und iterative Optimierungen durchzuführen, wobei ganze Systeme als individuelle Komponenten betrachtet werden können.

Das bedeutet, dass in einem Packingframework die Möglichkeit für die Umsetzung von Multi-Loop-Systemen bestehen muss. In *5.2.1 Packing als Teil der Gesamtsystemintegration* konnte mithilfe des vorgestellten Software-Stacks als Multi-Loop-System gezeigt werden, dass eine Aufteilung in iterativ zu lösende Teilprobleme ohne Weiteres möglich ist. Die Unterstützung der Lösung von aufeinander folgenden Teilproblemen ist aber auch deshalb notwendig, weil sich im klassischen Systementwurf eine intuitive Reihenfolge von Aufgaben etabliert hat, die es digital abzubilden und nacheinander zu lösen gilt. Diese stringente Einteilung des Gesamtentwurfs in einzelne Teilmodelle und deren Lösungsreihenfolge wird durch das Vorhandensein von leistungsfähigen digitalen Lösungen zur Unterstützung des Systementwurfs aber immer weiter aufgelöst. Auch in diesem Kontext muss die flexible Aufteilung in Teilmodelle oder Zusammenführung in ein Gesamtmodell gewährleistet sein. In den Beispielen aus *5.2 Packing in einem Flugzeugtragflügel* und *5.3 Packing und Piping auf einem Satellitenantriebspaneel* wurden dieselben Aufgaben - der Packingprozess und der Verrohrungsprozess einmal getrennt betrachtet und einmal in ein Gesamtmodell überführt. Während in ersterem Beispiel die Berechnung der Problemmodelle aufeinander erfolgte, wurde die Verrohrung der Bauteile im letzteren Beispiel in die Positionierung dergleichen integriert und parallel gelöst. Dadurch konnten theoretisch erst zu einem späteren Zeitpunkt zu untersuchende Aspekte bereits frühzeitig einfließen, wodurch eine globalere Betrachtung der Aufgabenstellung ermöglicht wurde.

Ob nun als ein umfassendes Gesamtproblem oder aufgespalten in einzelne Teilmodelle, ein Packingframework muss hinreichend flexibel sein, um beide Varianten darzustellen und zu verarbeiten. Hinsichtlich der Optimierung kann im Allgemeinen davon ausgegangen werden, dass die Rechenzeit zur Lösung eines Packingproblems von der Anzahl der zu optimierenden Kriterien abhängt. Um diese in akzeptablen Bereichen zu halten, sollte auch der Grad der Optimierung flexibel anpassbar sein. Zum einen sollte hierfür zwischen Optimierungskriterien variiert werden können, zum anderen wäre es wünschenswert, flexibel zwischen Optimierung und Entwurfsraumexploration wechseln zu können. Dies wird auch dadurch begründet, dass die Formalisierung der betrachteten Kriterien nicht immer möglich ist, sodass im Nachhinein doch subjektive Einschätzungen des Ingenieurs zur Auswahl einer Variante führen sollen. In *5.1 Gebäudepacking und -layouting in der Architektur* wurde die Eingliederung der automatisierten Packingoptimierung in eine Variantengenerierung vorgestellt. Dabei lag der Fokus auf der Fähigkeit der Packingentwurfssprache den Umfang der Optimierung innerhalb des Packingprozesses problemabhängig anzupassen und damit ein Wechselspiel zwischen einer Optimierung und einer Entwurfsraumexploration zu unterstützen. Weiterhin konnte demonstriert werden, dass auch nicht formalisierbare Kriterien keine weitestgehende Einschränkung darstellen, denn diese wurden parallel durch eine Exploration des Entwurfsraumes untersucht. Die Kombination aus der Optimierung von formalisierbaren Kriterien und der Entwurfsraumexploration für nicht formalisierbare Kriterien bietet einen weiteren Vorteil. Durch eine vorab getätigte Optimierung kann die abschließende Entscheidungsfindung durch den Experten bzw. die Expertin stark vereinfacht werden, da hierdurch eine gegebene Anzahl an (formalisierbaren) Kriterien auf eine Gesamtbewertung zusammenfällt, was die Dimensionalität des noch manuell zu bewertenden Entwurfsraumes reduziert. Mit *5.1 Gebäudepacking und -layouting in der Architektur* wurde ein Beispiel dafür gezeigt.

Integrationsfähigkeit Für ein erfolgreiches automatisiertes Layout- und Packingverfahren ist eine komfortable Integrationsfähigkeit in den Gesamtentwurf vonnöten. Dies betrifft zum einen, dass bereits vorliegende Methoden oder Prozesse weiterhin genutzt werden können, indem sie mit dem Layout- oder Packingprozess gekoppelt werden. Zum anderen muss der Prozess derart eingegliedert werden können, dass sowohl das Ergebnis der vorherigen Entwurfsschritte reibungslos eingelesen als auch das Ergebnis des Packings an die folgenden Entwurfsschritte

weiter gegeben werden kann, unter Umständen auch unter Ermöglichung von Rückkopplungen zwischen den einzelnen Teilschritten. Diese Integrationsfähigkeit konnte für das entwickelte Framework in den Beispielen illustriert werden. In *5.3 Packing und Piping auf einem Satellitenantriebspaneel* beispielsweise bestand bereits der digitale Auslegungsprozess für Satellitenpaneele bis zur Generierung der topologischen Struktur und aller geometrischer Satellitenantriebskomponenten. Das Packing der Bauteile und der Verrohrung wurde in diesen bestehenden Gesamtprozess mühelos integriert. In *5.2 Packing in einem Flugzeugtragflügel* dagegen wurde das Packingverfahren für die Positionierung von elektronischen Verteilern mit bereits bestehenden Verfahren für deren Verrohrung und Verkabelung gekoppelt, wodurch ein iterativer Systemauslegungsprozess entstand, welcher erfolgreich für eine Abschätzung der optimalen Tragflügelkonfiguration verwendet werden konnte.

Domänenunabhängigkeit & Visualisierungsfähigkeit Die Notwendigkeit der Domänenunabhängigkeit wird dadurch begründet, dass Layout- oder Packingprobleme in zahlreichen, teilweise sogar als disjunkt erscheinenden Bereichen existieren. Die in der vorliegenden Arbeit ausgewählten Anwendungsbeispiele entstammen unterschiedlichsten Fachgebieten wie dem Luftfahrtingenieurwesen, dem Raumfahrtingenieurwesen und dem Bauwesen bzw. der Architektur. Die Vielfalt der gestellten und diskutierten Fragestellungen konnte dabei zeigen, dass das entwickelte Framework durchaus unabhängig von der jeweiligen Domäne Anwendung finden konnte. Die Visualisierungsfähigkeit wurde auf ganz natürliche Weise in allen der Beispiele aufgezeigt, da in jeder der genannten Anwendungen die Ergebnisse nicht nur als Zahlenwerte ausgegeben, sondern auch in Form von geometrischen Modellen visualisiert wurden.

Unterstützung der Layout- und Packingaufgabe Ein weiterer zentraler Punkt der vorliegenden Arbeit ist die Frage, inwieweit die Modellierung von Packingprozessen durch das entwickelte Framework vereinfacht wird. In *5.1.2 Packingframework zur Layoutoptimierung im Detail* wurde diese Frage anhand einer nachfolgenden Grundrissplanung innerhalb eines Gebäudes diskutiert. Dafür wurde eine Reihe an Nebenbedingungen aus dem entwickelten Framework eingesetzt und im Detail aufgezeigt, wie diese als mathematisches Optimierungsproblem intern repräsentiert werden. Die Nebenbedingungen umfassten die Kollisionsfreiheit, die Inklusionsbedingungen sowie die Adjazenz- und Kontaktbedingungen und wurden durch Aufruf der jeweiligen Methoden durch das Framework automatisiert als mathematisches Modell erstellt. Auch die zur Verfügung gestellte Zielfunktion zur Aggregation der Einzelziele konnte ohne zusätzlichen Modellierungsaufwand zum Einsatz kommen. Der manuelle Definitionsaufwand darüber hinaus umfasste lediglich die Formulierung von Zusatzbedingung, welche nicht im Framework zur Verfügung stehen. Insgesamt wurde in dem Beispiel deutlich, dass nur wenige Schritte bei der Programmierung des eigentlichen Optimierungsmodells tatsächlich von dem/der Programmierer/-in übernommen werden müssen und die meisten der Teilmodelle durch den Einsatz der Methoden aus dem Packingframework für die manuelle Implementierung entfallen.

6.2 Ausblick

In der vorliegenden Arbeit wurde mithilfe von graphenbasierten Entwurfssprachen ein Framework entwickelt, welches das automatisierte Layouting und Packing von vernetzten Systemen unterstützt. Bei der Anwendung wurde dessen prinzipielle Funktionsfähigkeit nachgewiesen, dabei sind jedoch in höherer Detailtiefe auch weitere Diskussionspunkte aufgekommen. Der

erste Punkt betrifft die korrekte Auswahl eines geeigneten Optimierungsverfahrens zur Lösung der modellierten Aufgabenstellungen. Das entwickelte Framework erlaubt drei Optimierungsvarianten, die mathematische Optimierung durch Aufruf einer unabhängigen Lösungssoftware, die metaheuristische Optimierung mithilfe des integrierten Partikel-Multi-Schwarm-Verfahrens und eine Optimierung von extern durch den Einsatz einer Optimierungsplattform. Es konnte gezeigt werden, dass jedes der Verfahren zur Lösung von Packingproblemen eingesetzt werden kann, es stellt sich jedoch hierbei die Frage, welches Verfahren für welche Art der Problemstellung besonders geeignet ist. Ausgiebige Untersuchungsreihen mit der Fragestellung, welche Problemmodelle mit welchen Optimierungsverfahren besonders gut harmonieren oder welche Parametereinstellungen für eine Optimierung in Kombination mit einer bestimmten Problemstellung besonders effizient sind, konnten im Rahmen dieser Arbeit nicht durchgeführt werden. Von Interesse wäre außerdem auch eine Abschätzung, ob es eventuell sogar ein einzelnes Optimierungsverfahren gibt, das allen anderen Varianten grundsätzlich überlegen ist. Auch dafür kann diese Arbeit keine endgültige Antwort finden, das entwickelte Framework bietet aber eine gute Basis für eine Reihe ausführlicher weiterer Untersuchungen.

Eine weitere Erkenntnis, welche während der Erstellung und Ausführung der Anwendungsbeispiele gewonnen werden konnte, betrifft die Verarbeitungsgeschwindigkeit. Vor allem bei der vereinfachten gesamtheitlichen Auslegung eines Flugzeugtragflügels ist deutlich geworden, dass zwar eine automatisierte ganzheitliche Systemoptimierung durch eine reine Iteration von aufeinanderfolgenden Teilverfahren möglich ist, dies aber eine schier unendlich große Vielfalt an zu untersuchenden Varianten zur Folge hat. Eine ähnliche Beobachtung wurde bei der Auslegung des Satellitenantriebspaneels unter Verwendung der mathematischen Optimierung gemacht, bei der die Laufzeiten mit der Systemgröße enorm anstiegen. In Zukunft müssen Möglichkeiten gefunden werden, die einzelnen Teilaufgaben zu beschleunigen, um in akzeptabler Zeit auch Ergebnisse für komplexere Systeme zu erhalten. Mögliche Verbesserungen könnten eventuell durch weitere Parallelisierung erzielt werden oder durch die Auslagerung der Berechnung auf Hochleistungsrechensysteme.

Zukünftige Aufgaben sollten sich auch der weiteren Anreicherung der zur Verfügung gestellten Modelle über die geometrischen Aspekte hinaus widmen, um physikalische Randbedingungen aus den Bereichen Mechanik, Thermodynamik, Hydraulik, Optik und vielen weiteren zu integrieren. Neben der reinen Erweiterung um neue Modelle ist außerdem die Ersetzung von bereits implementierten Nebenbedingungen durch effizientere oder numerisch genauere Varianten vorstellbar. Auch die Repräsentation von weiteren komplexeren Freiheitsgraden stellt eine relevante Forschungsfrage dar, wenn zukünftig reale Probleme aus der Industrie gelöst werden sollen. Hinsichtlich der Gestalt der Bauteilgeometrien sind im entwickelten Framework z.B. bisher lediglich Freiheitsgrade für die Skalierung vorgesehen. Interessant wären aber auch komplette Freiformänderungen der Geometrie, um grundsätzlich in der Gestalt variable Bauteile unterzubringen oder den Bauraum flexibel anzupassen.

Alles in allem bleiben also zahlreiche spannende Anknüpfungspunkte für weitere Untersuchungen, welche mithilfe des entwickelten Frameworks durchgeführt werden können sowie Vorschläge für dessen Erweiterung und Ausbau. Aufgrund der gegenwärtigen und sehr wahrscheinlich auch noch die Zukunft prägenden Entwicklung, in der eine immer lautere Forderung nach der Steigerung der Energieeffizienz bei gleichzeitiger Reduktion der Emissionen sowie der Stärkung der wirtschaftlichen Wettbewerbsfähigkeit in nahezu allen technischen Bereichen zu beobachten ist, ist die Schaffung von effektiven Methoden zur Entwurfsautomatisierung von Layout- und Packingaufgaben auch weiterhin unerlässlich. Im Kontext der graphenbasierten Entwurfssprachen wurden in der vorliegenden Arbeit hierfür die ersten Grundsteine gelegt, welche eine vielversprechende Zukunft weisen.

Literatur

[Aladahalli et al., 2005]
Chandankumar Aladahalli, Jonathan Cagan und Kenji Shimada (Dez. 2005). „Objective Function Effect Based Pattern Search—An Implementation for 3D Component Layout“. In: *Journal of Mechanical Design* 129.3, S. 255–265. ISSN: 1050-0472. DOI: 10.1115/1.2406096. URL: https://doi.org/10.1115/1.2406096.

[Aladahalli et al., 2007]
Chandankumar Aladahalli, Jonathan Cagan und Kenji Shimada (März 2007). „Objective Function Effect Based Pattern Search—Theoretical Framework Inspired by 3D Component Layout“. In: *Journal of Mechanical Design - J MECH DESIGN* 129. DOI: 10.1115/1.2406095.

[Allahyari und Azab, 2018]
Maral Zafar Allahyari und Ahmed Azab (2018). „Mathematical modeling and multi-start search simulated annealing for unequal-area facility layout problem“. In: *Expert Systems with Applications* 91, S. 46–62. ISSN: 0957-4174. DOI: https://doi.org/10.1016/j.eswa.2017.07.049.

[E. Andersen und K. Andersen, 1999]
Erling Andersen und Knud Andersen (Jan. 1999). „The Mosek Interior Point Optimizer for Linear Programming: An Implementation of the Homogeneous Algorithm“. In: 33. DOI: 10.1007/978-1-4757-3216-0_8.

[E. Andersen et al., 2003]
Erling Andersen, Cornelis Roos und Tamás Terlaky (Feb. 2003). „On implementing a primal-dual interior-point method for conic quadratic programming“. In: *Mathematical Programming* 95, S. 249–277. DOI: 10.1007/s10107-002-0349-3.

[E. Andersen und Ye, 1997]
Erling Andersen und Yinyu Ye (Dez. 1997). „A Computational Study of the Homogeneous Algorithm for Large-Scale Convex Optimization“. In: *Computational Optimization and Applications* 10(3). DOI: 10.1023/A:1018369223322.

[Araújo et al., 2019]
Luiz J.P. Araújo, Ender Özcan, Jason A. D. Atkin und Martin Baumers (2019). „Analysis of irregular three-dimensional packing problems in additive manufacturing: a new taxonomy and dataset“. In.

[Bai und Rubin, 2009]
Lihui Bai und Paul A. Rubin (2009). „Combinatorial Benders Cuts for the Minimum Tollbooth Problem“. In: *Operations Research* 57.6, S. 1510–1522. DOI: https://doi.org/10.1287/opre.1090.0694.

[Beghini et al., 2014]
Lauren L. Beghini, Alessandro Beghini, Neil Katz, William F. Baker und Glaucio H. Paulino (2014). „Connecting architecture and engineering through structural topology opti-

mization“. In: *Structural Engineering Review* 59, S. 716–726. DOI: 10.1016/j.engstruct.2013.10.032.

[Belotti et al., 2016]
Pietro Belotti, Pierre Bonami, Matteo Fischetti, Andrea Lodi, Michele Monaci, Amaya Nogales-Gómez und Domenico Salvagnin (Dez. 2016). „On handling indicator constraints in mixed integer programming“. In: *Computational Optimization and Applications* 65. DOI: 10.1007/s10589-016-9847-8.

[Belotti et al., 2009]
Pietro Belotti, Jon Lee, Leo Liberti, François Margot und Andreas Wächter (Okt. 2009). „Branching and bounds tightening techniques for non-convex MINLP“. In: *Optimization Methods and Software* 24, S. 597–634. DOI: 10.1080/10556780903087124.

[Belov et al., 2018]
G. Belov, T. Czauderna, M. G. D. L. Banda, Matthias Klapperstück, Ilankaikone Senthooran, Mitch Smith, Michael Wybrow und M. Wallace (2018). „Process Plant Layout Optimization: Equipment Allocation“. In: *CP*. DOI: 10.1007/978-3-319-98334-9_31.

[Berkelaar et al., 2004]
Michel Berkelaar, Kjell Eikland und Peter Notebaert (Mai 2004). *lp_solve 5.5, Open source (Mixed-Integer) Linear Programming system*. Software. Language : Multi-platform, pure ANSI C / POSIX source code, Lex/Yacc based parsing Official. Release data : Version 5.1.0.0 dated 1 May 2004. Last accessed Jan, 18 2020. URL: http://lpsolve.sourceforge.net/5.5/.

[Bortfeldt und Wäscher, 2013]
Andreas Bortfeldt und Gerhard Wäscher (Aug. 2013). „Constraints in container loading – A state-of-the-art review“. In: *European Journal of Operational Research* 229, S. 1–20. DOI: 10.1016/j.ejor.2012.12.006.

[Cagan et al., 2002]
Jonathan Cagan, Kenji Shimada und Sun Yin (2002). „A survey of computational approaches to three-dimensional layout problems“. In: *Computer-Aided Design* 34.8, S. 597–611. ISSN: 0010-4485. DOI: https://doi.org/10.1016/S0010-4485(01)00109-9.

[Carta, 2020]
Silvio Carta (2020). *Algorithms are designing better buildings*. Zuletzt aufgerufen: 20.03.2022.

[Carta et al., 2020]
Silvio Carta, Stephanie St. Loe, Tommaso Turchi und Joel Simon (2020). „Self-Organising Floor Plans in Care Homes“. In: *Sustainability* 12.11. ISSN: 2071-1050. DOI: 10.3390/su12114393. URL: https://www.mdpi.com/2071-1050/12/11/4393.

[Chernov et al., 2010]
N. Chernov, Yu. Stoyan und T. Romanova (2010). „Mathematical model and efficient algorithms for object packing problem“. In: *Computational Geometry* 43.5, S. 535–553. ISSN: 0925-7721. DOI: https://doi.org/10.1016/j.comgeo.2009.12.003.

[Cherri et al., 2019]
Luiz Henrique Cherri, Maria Carravilla, Cristina Ribeiro und Franklina Toledo (Sep. 2019). „Optimality in nesting problems: New constraint programming models and a new global constraint for non-overlap“. In: *Operations Research Perspectives* 6, S. 100125. DOI: 10.1016/j.orp.2019.100125.

[Cherri et al., 2018]
Luiz Henrique Cherri, Adriana Cherri und Edilaine Soler (Sep. 2018). „Mixed integer quadratically-constrained programming model to solve the irregular strip packing problem with continuous rotations“. In: *Journal of Global Optimization* 72, S. 1–19. DOI: 10.1007/s10898-018-0638-x.

[CleanAviation Media, 2022]
CleanAviation Media (2022). *Physical Architecture Optimization System (PHAROS)*. PHAROS Project Article, https://www.clean-aviation.eu/media/news/simplifying-the-3d-packaging-3d-piping-and-3d-routing-sequence-for-physical-system-architecture, Last Access February 27, 2022. URL: https://cordis.europa.eu/project/id/865044.

[CleanSky2, 2022]
CleanSky2 (2022). *Physical Architecture Optimization System (PHAROS)*. Project Description, https://cordis.europa.eu/project/id/865044, Last Access February 27, 2022. URL: https://cordis.europa.eu/project/id/865044.

[Codato und Fischetti, 2006]
Gianni Codato und Matteo Fischetti (Aug. 2006). „Combinatorial Benders' Cuts for Mixed-Integer Linear Programming". In: *Operations Research* 54, S. 756–766. DOI: 10.1287/opre.1060.0286.

[Cohen et al., 1995]
Jonathan Cohen, Ming Lin, Dinesh Manocha und Madhav Ponamgi (März 1995). „I-COLLIDE: An Interactive and Exact Collision Detection System for Large-Scale Environments". In: *Proceedings of ACM Interactive 3D Graphics Conference* 1. DOI: 10.1145/199404.199437.

[Corney und Lim, 2002]
Jonathan R. Corney und Theodore Lim (2002). „3D Modeling with ACIS". In.

[Dandurand et al., 2014]
B. Dandurand, P. Guarneri, G. Fadel und M. Wiecek (2014). „Bilevel multiobjective packaging optimization for automotive design". In: *Structural and Multidisciplinary Optimization* 50, S. 663–682.

[Demiröz et al., 2019]
Barış Evrim Demiröz, İ. Kuban Altınel und Lale Akarun (2019). „Rectangle blanket problem: Binary integer linear programming formulation and solution algorithms". In: *European Journal of Operational Research* 277.1, S. 62–83. URL: https://ideas.repec.org/a/eee/ejores/v277y2019i1p62-83.html.

[Dinkelacker et al., 2021]
J. Dinkelacker, D. Kaiser, M. Panzeri, P. Parmentier, M. Neumaier, C. Tonhäuser und S Rudolph (2021). „System Integration based on packaging, piping and harness routing automation using graph-based design languages". In: *Deutscher Luft- und Raumfahrtkongress 2021*. August 31 - September 2.

[Dong et al., 2011]
Hong Dong, Paolo Guarneri und Georges Fadel (Mai 2011). „Bi-level Approach to Vehicle Component Layout With Shape Morphing". In: *Journal of Mechanical Design* 133.4. 041008. ISSN: 1050-0472. DOI: 10.1115/1.4003916. URL: https://doi.org/10.1115/1.4003916.

[Drira et al., 2007]
Amine Drira, Henri Pierreval und Sonia Hajri-Gabouj (2007). „Facility layout problems: A survey". In: *Annual reviews in control* 31.2, S. 255–267.

[Edelkamp und Wichern, 2015]
Stefan Edelkamp und Paul Wichern (2015). „Packing Irregular-Shaped Objects for 3D Printing". In: *KI 2015: Advances in Artificial Intelligence*. Hrsg. von Steffen Hölldobler, Rafael Peñaloza und Sebastian Rudolph, S. 45–58.

[Ericson, 2004]
Christer Ericson (2004). *Real-time collision detection*. CRC Press.

[Fadel und Wiecek, 2015]
Georges Fadel und Margaret Wiecek (Jan. 2015). „Packing Optimization of Free-Form

Objects in Engineering Design“. In: S. 37–66. ISBN: 978-3-319-18898-0. DOI: 10.1007/978-3-319-18899-7_3.

[Fan et al., 2016]
Zhun Fan, Hui Li, Caimin Wei, Wenji Li, Han Huang, Xinye Cai und Zhaoquan Cai (Dez. 2016). „An improved epsilon constraint handling method embedded in MOEA/D for constrained multi-objective optimization problems“. In: S. 1–8. DOI: 10.1109/SSCI.2016.7850224.

[G. Fasano, 2015]
G. Fasano (2015). „A Modeling-Based Approach for Non-standard Packing Problems“. In: Springer, S. 67–85.

[Giorgio Fasano, 2013]
Giorgio Fasano (Feb. 2013). „A global optimization point of view to handle non-standard object packing problems“. In: *Journal of Global Optimization* 55. DOI: 10.1007/s10898-012-9865-8.

[Giorgio Fasano, 2014]
Giorgio Fasano (Jan. 2014). *Solving Non-standard Packing Problems by Global Optimization and Heuristics*. ISBN: 978-3-319-05004-1. DOI: 10.1007/978-3-319-05005-8.

[Giorgio Fasano et al., 2014]
Giorgio Fasano, Cristina Gastaldi, Annamaria Piras und Dario Saia (2014). „An optimization framework to tackle challenging cargo accommodation tasks in space engineering“. In.

[Giorgio Fasano et al., 2015]
Giorgio Fasano, Claudia Lavopa, Davide Negri und Maria Vola (Jan. 2015). „CAST: A Successful Project in Support of the International Space Station Logistics“. In: Bd. 105, S. 87–117. ISBN: 978-3-319-18898-0. DOI: 10.1007/978-3-319-18899-7_5.

[Gamma et al., 1995]
Erich Gamma, Richard Helm, Ralph Johnson und John M. Vlissides (1995). *Design patterns: elements of reusable object-oriented software*. Addison-Wesley professional computing series. Reading, Mass. [u.a.]: Addison-Wesley. ISBN: 9780201633610.

[Gamma et al., 1996]
Erich Gamma, Richard Helm, Ralph E. Johnson und John Vlissides (1996). *Entwurfsmuster: Elemente wiederverwendbarer objektorientierter Software*. München: Addison-Wesley. ISBN: 978-3-89319-950-1.

[Gamrath et al., 2020]
Gerald Gamrath, Daniel Anderson, Ksenia Bestuzheva, Wei-Kun Chen, Leon Eifler, Maxime Gasse, Patrick Gemander, Ambros Gleixner, Leona Gottwald, Katrin Halbig, Gregor Hendel, Christopher Hojny, Thorsten Koch, Pierre Le Bodic, Stephen J. Maher, Frederic Matter, Matthias Miltenberger, Erik Mühmer, Benjamin Müller, Marc E. Pfetsch, Franziska Schlösser, Felipe Serrano, Yuji Shinano, Christine Tawfik, Stefan Vigerske, Fabian Wegscheider, Dieter Weninger und Jakob Witzig (März 2020). *The SCIP Optimization Suite 7.0*. Technical Report. Optimization Online. URL: http://www.optimization-online.org/DB_HTML/2020/03/7705.html.

[Gärtner, 1999]
Bernd Gärtner (1999). „Fast and Robust Smallest Enclosing Balls“. In: *ESA*.

[Ge et al., 2021]
Dongdong Ge, Chengwen Chris Wang, Zikai Xiong und Yinyu Ye (2021). „From an Interior Point to a Corner Point: Smart Crossover“. In.

[Geißler et al., 2012]
Björn Geißler, Alexander Martin, Antonio Morsi und Lars Schewe (Nov. 2012). „Using

Piecewise Linear Functions for Solving MINLPs“. In: Bd. 154, S. 287–314. ISBN: 978-1-4614-1926-6. DOI: 10.1007/978-1-4614-1927-3_10.

[Gliozzi et al., 2015]
Stefano Gliozzi, Alessandro Castellazzo und Giorgio Fasano (Jan. 2015). „A container loading problem MILP-based heuristics solved by CPLEX: An experimental analysis“. In: Bd. 105, S. 157–173. DOI: 10.1007/978-3-319-18899-7_7.

[Gliozzi et al., 2016]
Stefano Gliozzi, Alessandro Castellazzo und Giorgio Fasano (2016). „Packing Problems in Space Solved by CPLEX: An Experimental Analysis“. In: Hrsg. von Giorgio Fasano und János D. Pintér. Cham: Springer International Publishing, S. 129–150. ISBN: 978-3-319-41508-6. DOI: 10.1007/978-3-319-41508-6_5. URL: https://doi.org/10.1007/978-3-319-41508-6_5.

[*GNU Linear Programming Kit, Version 4.65* 2018]
GNU Linear Programming Kit, Version 4.65 (2018). URL: http://www.gnu.org/software/glpk/glpk.html.

[Grebennik et al., 2018]
I. Grebennik, A. Kovalenko, T. Romanova, I. Urniaieva und S. Shekhovtsov (2018). „Combinatorial Configurations in Balance Layout Optimization Problems“. In: *Cybernetics and Systems Analysis* 54, S. 221–231.

[Grignon und Fadel, 2004]
Pierre M. Grignon und Georges Fadel (2004). „A GA based configuration design optimization method“. In: *J. Mech. Des.* 126.1, S. 6–15.

[Griva et al., 2008]
Igor Griva, Stephen G. Nash und Ariela Sofer (2008). *Linear and Nonlinear Optimization (2. ed.)*. SIAM, S. I–XXII, 1–742. ISBN: 978-0-89871-661-0.

[Groell, 2018]
Lutz Groell (Nov. 2018). „Klassifikation von Optimierungsproblemen“. In: *at - Automatisierungstechnik* 66, S. 903–927. DOI: 10.1515/auto-2018-0081.

[Guo und Li, 2017]
Zifeng Guo und Biao Li (2017). „Evolutionary approach for spatial architecture layout design enhanced by an agent-based topology finding system“. In: *Frontiers of Architectural Research* 6.1, S. 53–62. ISSN: 2095-2635. DOI: https://doi.org/10.1016/j.foar.2016.11.003. URL: https://www.sciencedirect.com/science/article/pii/S2095263516300565.

[Gurobi Optimization, LLC, 2022a]
Gurobi Optimization, LLC (2022a). *Gurobi Optimizer Reference Manual*. URL: https://www.gurobi.com.

[Gurobi Optimization, LLC, 2022b]
Gurobi Optimization, LLC (2022b). *The Most Advanced Algorithms*. 2. März 2011. Abgerufen am 07.02.2022. URL: https://www.gurobi.com/products/gurobi-optimizer/.

[Hamacher und Klamroth, 2006]
Horst W. Hamacher und Kathrin Klamroth (2006). *Lineare Optimierung und Netzwerkoptimierung: zweisprachige Ausgabe Deutsch Englisch*. 2., verb. Aufl. Studium. Wiesbaden: Vieweg. ISBN: 978-3-8348-0185-2. URL: http://digitale-objekte.hbz-nrw.de/storage/2007/05/16/file_226/1810609.pdf.

[Helwig et al., 2013]
Sabine Helwig, Juergen Branke und Sanaz Mostaghim (Apr. 2013). „Experimental Analysis of Bound Handling Techniques in Particle Swarm Optimization“. In: *IEEE Transactions on Evolutionary Computation* 17, S. 259–271. DOI: 10.1109/TEVC.2012.2189404.

[Hochstättler, 2010]
Winfried Hochstättler (2010). *Algorithmische Mathematik*. Springer-Lehrbuch. Berlin Heidelberg: Springer. ISBN: 9783642054211.

[IBM Academic Initiative, 2020]
IBM Academic Initiative (2020). Abgerufen am 04.02.2022. URL: https://www.ibm.com/academic/home.

[IBM ILOG CPLEX, 2019]
IBM ILOG CPLEX (2019). *CPLEX Optimization Studio 12.10.0*. URL: https://www.ibm.com/docs/en/icos/12.10.0.

[IILS, 2022]
IILS (2022). *Ingenieurgesellschaft für Intelligente Lösungen und Systeme mbH*. URL: https://www.iils.de/.

[Jacobson, 2017]
Alec Jacobson (2017). „Generalized matryoshka: Computational design of nesting objects". In: *Computer Graphics Forum* 36.5, S. 27–35.

[Jacquenot et al., 2009]
Guillaume Jacquenot, Fouad Bennis, Jean-Jacques Maisonneuve und Philippe Wenger (2009). „2d multi-objective placement algorithm for free-form components". In: *ASME 2009 International Design Engineering Technical Conferences and Computers and Information in Engineering Conference*, S. 239–248.

[Jessee et al., 2020]
Alex Jessee, Satya R. T. Peddada, Danny J. Lohan, James T. Allison und Kai A. James (Mai 2020). „Simultaneous Packing and Routing Optimization Using Geometric Projection". In: *Journal of Mechanical Design* 142.11. 111702. ISSN: 1050-0472. DOI: 10.1115/1.4046809. URL: https://doi.org/10.1115/1.4046809.

[Joung und Do Noh, 2014]
Youn-Kyoung Joung und Sang Do Noh (2014). „Intelligent 3D packing using a grouping algorithm for automotive container engineering". In: *Journal of Computational Design and Engineering* 1.2, S. 140–151. ISSN: 2288-4300. DOI: https://doi.org/10.7315/JCDE.2014.014. URL: https://www.sciencedirect.com/science/article/pii/S228843001450019X.

[Katragadda et al., 2012]
Predicting the Thermal Performance for the Multi-Objective Vehicle Underhood Packing Optimization Problem (Aug. 2012). Bd. Volume 3: 38th Design Automation Conference, Parts A and B. International Design Engineering Technical Conferences and Computers and Information in Engineering Conference, S. 745–752. DOI: 10.1115/DETC2012-71098. URL: https://doi.org/10.1115/DETC2012-71098.

[Larsson und Källberg, 2013]
Thomas Larsson und Linus Källberg (Aug. 2013). „Fast and Robust Approximation of Smallest Enclosing Balls in Arbitrary Dimensions". In: *Computer Graphics Forum* 32. DOI: 10.1111/cgf.12176.

[Laskari et al., 2002]
Elena C. Laskari, Konstantinos E. Parsopoulos und Michael N. Vrahatis (2002). „Particle swarm optimization for integer programming". In: *Proceedings of the 2002 Congress on Evolutionary Computation. CEC'02 (Cat. No.02TH8600)* 2, 1582–1587 vol.2.

[S. Lee und K. S. Lee, 2020]
Sewon Lee und Kyung Sun Lee (2020). „Optimization of Apartment-Complex Layout Planning for Daylight Accessibility in a High-Density City with a Temperate Climate". In: *Energies* 13.16. ISSN: 1996-1073. DOI: 10.3390/en13164172. URL: https://www.mdpi.com/1996-1073/13/16/4172.

[Linderoth und Luedtke, 2016]
Jeff Linderoth und Jim Luedtke (2016). „Mixed Integer Nonlinear Programming“. Vorlesungsunterlagen. Wisconsin-Madison: Department of Industrial und Systems Engineering, University of Wisconsin-Madison. URL: https://www.ima.umn.edu/materials/2015-2016/ND8.1-12.16/25419/Luedtke-minlp.pdf.

[Jingfa Liu und Jun Liu, 2019]
Jingfa Liu und Jun Liu (2019). „Applying multi-objective ant colony optimization algorithm for solving the unequal area facility layout problems“. In: *Applied Soft Computing* 74, S. 167–189. ISSN: 1568-4946. DOI: https://doi.org/10.1016/j.asoc.2018.10.012. URL: https://www.sciencedirect.com/science/article/pii/S1568494618305696.

[X. Liu et al., 2015]
Xiao Liu, Jia-min Liu, An-xi Cao und Zhuang-le Yao (Mai 2015). „HAPE3D—a new constructive algorithm for the 3D irregular packing problem“. In: *Frontiers of Information Technology & Electronic Engineering* 16, S. 380–390. DOI: 10.1631/FITEE.1400421.

[Lougee, 2003]
Robin Lougee (Feb. 2003). „The Common Optimization Interface for Operations Research: Promoting open-source software in the operations research community“. In: *IBM Journal of Research and Development* 47, S. 57–66. DOI: 10.1147/rd.471.0057.

[Lutters et al., 2012]
E. Lutters, D. ten Dam und T. Faneker (2012). „3D Nesting of Complex Shapes“. In: *Procedia CIRP* 3. 45th CIRP Conference on Manufacturing Systems 2012, S. 26–31. ISSN: 2212-8271. DOI: https://doi.org/10.1016/j.procir.2012.07.006. URL: http://www.sciencedirect.com/science/article/pii/S2212827112001783.

[Ma et al., 2018]
Y. Ma, Z. Chen, W. Hu und W. Wang (Aug. 2018). „Packing Irregular Objects in 3D Space via Hybrid Optimization“. In: *Computer Graphics Forum* 37, S. 49–59. DOI: 10.1111/cgf.13490.

[Magalhaes-Mendes et al., 2017]
Jorge Magalhaes-Mendes, Pinheiro Rogério Plácido und Amaro Júnior Bonfim (2017). „A Parallel Biased Random-Key Genetic Algorithm with Multiple Populations Applied to Irregular Strip Packing Problems“. In: *Mathematical Problems in Engineering* 2017. DOI: 10.1155/2017/1670709.

[Mamou und Ghorbel, 2009]
Khaled Mamou und Faouzi Ghorbel (Nov. 2009). „A simple and efficient approach for 3D mesh approximate convex decomposition“. In: *Proceedings - International Conference on Image Processing, ICIP*, S. 3501–3504. DOI: 10.1109/ICIP.2009.5414068. URL: https://github.com/kmammou/v-hacd.

[McKendall und Hakobyan, 2021]
Alan McKendall und Artak Hakobyan (2021). „An Application of an Unequal-Area Facilities Layout Problem with Fixed-Shape Facilities“. In: *Algorithms* 14.11. ISSN: 1999-4893. DOI: 10.3390/a14110306. URL: https://www.mdpi.com/1999-4893/14/11/306.

[MOSEK ApS, 2019]
MOSEK ApS (2019). *MOSEK Optimizer API for Java 9.3.13*. URL: https://docs.mosek.com/latest/javaapi/index.html.

[Noesis Solutions, 2022]
Noesis Solutions (2022). *Optimus*. URL: https://www.noesissolutions.com/.

[Panesar et al., 2015]
Ajit Panesar, David Brackett, Ian Ashcroft, Ricky Wildman und Richard Hague (Juli 2015). „Design Framework for Multifunctional Additive Manufacturing: Placement and

Routing of 3D Printed Circuit Volumes“. In: *Journal of Mechanical Design* 137. DOI: 10.1115/1.4030996.

[Paquay et al., 2014]
C. Paquay, Michael Schyns und Sabine Limbourg (Juli 2014). „A mixed integer programming formulation for the three-dimensional bin packing problem deriving from an air cargo application“. In: *International Transactions in Operational Research* 23. DOI: 10.1111/itor.12111.

[Peddada et al., 2021a]
Satya Peddada, Kai James und James Allison (März 2021a). „A Novel Two-Stage Design Framework for Two-Dimensional Spatial Packing of Interconnected Components“. In: *Journal of Mechanical Design* 143. DOI: 10.1115/1.4048817.

[Peddada et al., 2022]
Satya Peddada, Lawrence E. Zeidner, Horea T. Ilies, Kai A. James und James T. Allison (Aug. 2022). „Toward Holistic Design of Spatial Packaging of Interconnected Systems With Physical Interactions (SPI2)“. In: *Journal of Mechanical Design* 144.12, S. 120801. ISSN: 1050-0472. DOI: 10.1115/1.4055055. eprint: https://asmedigitalcollection.asme.org/mechanicaldesign/article-pdf/144/12/120801/6911953/md_144_12_120801.pdf. URL: https://doi.org/10.1115/1.4055055.

[Peddada et al., 2021b]
An Introduction to 3D SPI2 (Spatial Packaging of Interconnected Systems With Physics Interactions) Design Problems: A Review of Related Work, Existing Gaps, Challenges, and Opportunities (Aug. 2021b). Bd. Volume 3B: 47th Design Automation Conference (DAC). International Design Engineering Technical Conferences and Computers and Information in Engineering Conference. DOI: 10.1115/DETC2021-72106. URL: https://doi.org/10.1115/DETC2021-72106.

[Perron und Furnon, 2021]
Laurent Perron und Vincent Furnon (1. Okt. 2021). *OR-Tools*. Version 9.1. Google. URL: https://developers.google.com/optimization/.

[Pichugina und Yakovlev, 2020]
Oksana Pichugina und Sergiy Yakovlev (2020). „Euclidean Combinatorial Configurations: Continuous Representations and Convex Extensions“. In: *Lecture Notes in Computational Intelligence and Decision Making*. Hrsg. von Volodymyr Lytvynenko, Sergii Babichev, Waldemar Wójcik, Olena Vynokurova, Svetlana Vyshemyrskaya und Svetlana Radetskaya, S. 65–80.

[Ravindranath, 2011]
K. Ravindranath (2011). „Agent-based under hood packing“. In.

[Reichel, 2006]
Joachim Reichel (2006). „Combinatorial approaches for the trunk packing problem“. Diss. DOI: http://dx.doi.org/10.22028/D291-25883.

[Rockafellar, 1993]
R. Tyrrell Rockafellar (1993). „Lagrange Multipliers and Optimality“. In: *SIAM Review* 35.2, S. 183–238. ISSN: 00361445. URL: http://www.jstor.org/stable/2133143.

[Rodrigues et al., 2013a]
Eugénio Rodrigues, Adélio Rodrigues Gaspar und Álvaro Gomes (2013a). „An evolutionary strategy enhanced with a local search technique for the space allocation problem in architecture, Part 1: Methodology“. In: *Computer-Aided Design* 45.5, S. 887–897. ISSN: 0010-4485. DOI: https://doi.org/10.1016/j.cad.2013.01.001. URL: https://www.sciencedirect.com/science/article/pii/S0010448513000031.

[Rodrigues et al., 2013b]
Eugénio Rodrigues, Adélio Rodrigues Gaspar und Álvaro Gomes (2013b). „An evolutio-

nary strategy enhanced with a local search technique for the space allocation problem in architecture, Part 2: Validation and performance tests“. In: *Computer-Aided Design* 45.5, S. 898–910. ISSN: 0010-4485. DOI: https://doi.org/10.1016/j.cad.2013.01.003. URL: https://www.sciencedirect.com/science/article/pii/S0010448513000055.

[T. Romanova et al., 2018]
T. Romanova, J. Bennell, Y. Stoyan und A. Pankratov (2018). „Packing of concave polyhedra with continuous rotations using nonlinear optimisation“. In: *European Journal of Operational Research* 268.1, S. 37–53. ISSN: 0377-2217. DOI: https://doi.org/10.1016/j.ejor.2018.01.025.

[Rudolph, 2002]
Stephan Rudolph (2002). „Übertragung von Ähnlichkeitsbegriffen“. Habilitationsschrift. Institut für Statik- und Dynamik der Luft- und Raumfahrtkonstruktionen, Universität Stuttgart.

[Rudolph, 2019]
Stephan Rudolph (2019). *Physical Architecture Optimization System (PHAROS)*. Submitted February 2019. H2020 CS2 Grant Proposal, Part B.I (Confidential, Unpublished).

[Scheithauer et al., 2005]
Guntram Scheithauer, Yu Stoyan und Tatiana Romanova (Jan. 2005). „Mathematical Modeling of Interactions of Primary Geometric 3D Objects“. In: *Cybernetics and Systems Analysis* 41, S. 332–342. DOI: 10.1007/s10559-005-0067-y.

[Scheithauer und Terno, 1993]
Guntram Scheithauer und J. Terno (Jan. 1993). „Modeling of packing problems“. In: *Optimization* 28, S. 63–84. DOI: 10.1080/02331939308843904.

[Schewe und M. Schmidt, 2019]
Lars Schewe und Martin Schmidt (Jan. 2019). *Optimierung von Versorgungsnetzen: Mathematische Modellierung und Lösungstechniken*. ISBN: 978-3-662-58538-2. DOI: 10.1007/978-3-662-58539-9.

[J. Schmidt, 2017]
Jens Schmidt (2017). *Total Engineering Automation*. Hrsg. von IILS mbH. Zuletzt geprüft am 20.04.2022. URL: https://www.iils.de/#downloads.

[Schroeder et al., 2006]
William J. Schroeder, Ken Martin, William E. Lorensen, Lisa Sobierajski Avila und Kenneth W. Martin (2006). *The visualization toolkit. an object-oriented approach to 3D graphics ; [visualize data in 3D - medical, engineering or scientific ; build your own applications with C++, Tcl, Java or Python ; includes source code for VTK (supports UNIX, Windows and Mac)]*. eng. 4. ed. Literaturangaben. [Clifton Park, NY]: Kitware, XVI, 512 S. ISBN: 978-1-930934-19-1.

[Schwenkert und Stry, 2015]
Rainer Schwenkert und Yvonne Stry (2015). *Operations Research kompakt: Eine an Beispielen orientierte Einführung*. ger. 1. Aufl. 2015. Operations Research kompakt. Berlin: Gabler. ISBN: 9783662483978.

[SCIP Optimization Suite, 2022]
SCIP Optimization Suite (2022). *What is SCIP?* Abgerufen am 07.02.2022. URL: https://www.scipopt.org/index.php#about.

[Sergeyev et al., 2018]
Ya. D. Sergeyev, D. E. Kvasov und M. S. Mukhametzhanov (Jan. 2018). „On the efficiency of nature-inspired metaheuristics in expensive global optimization with limited budget“. In: *Scientific Reports* 8. DOI: 10.1038/s41598-017-18940-4.

[Sho, 2017]
Hiroshi Sho (2017). „Particle multi-swarm optimization: A proposal of multiple particle

swarm optimizers with information sharing“. In: *2017 IEEE 10th International Workshop on Computational Intelligence and Applications (IWCIA)*, S. 109–114. DOI: 10.1109/IWCIA.2017.8203570.

[Sörensen und Glover, 2013]
Kenneth Sörensen und Fred Glover (Jan. 2013). „Metaheuristics“. In: S. 960–970. ISBN: 978-1-4419-1137-7. DOI: 10.1007/978-1-4419-1153-7_1167.

[Sridhar et al., 2017]
R Sridhar, Dr.M. Chandrasekaran, C Sriramya und Tom Page (März 2017). „Optimization of heterogeneous Bin packing using adaptive genetic algorithm“. In: *IOP Conference Series: Materials Science and Engineering* 183, S. 012026. DOI: 10.1088/1757-899X/183/1/012026.

[Stein, 2018]
Oliver Stein (2018). *Grundzüge der Nichtlinearen Optimierung*. Springer Verlag. 218 S. ISBN: 978-3-662-55592-7. DOI: 10.1007/978-3-662-55593-4.

[Stein, 2021]
Oliver Stein (2021). *Grundzüge der globalen Optimierung*. Deutsch. 2. Auflage. Springer eBook Collection. Berlin ; Heidelberg: Springer Spektrum. ISBN: 9783662625347. URL: http://dx.doi.org/10.1007/978-3-662-62534-7.

[Y. Stoyan et al., 2016]
Y. Stoyan, T. Romanova, A. Pankratov, A. Kovalenko und P. Stetsyuk (2016). „Balance Layout Problems: Mathematical Modeling and Nonlinear Optimization“. In.

[Yuri Stoyan und Tatiana Romanova, 2013]
Yuri Stoyan und Tatiana Romanova (2013). „Mathematical Models of Placement Optimisation: Two- and Three-Dimensional Problems and Applications“. In: Hrsg. von Giorgio Fasano und János D. Pintér. New York, NY: Springer New York, S. 363–388. ISBN: 978-1-4614-4469-5. DOI: 10.1007/978-1-4614-4469-5_15. URL: https://doi.org/10.1007/978-1-4614-4469-5_15.

[Yurij Stoyan et al., 2004]
Yurij Stoyan, Guntram Scheithauer, Nikolay Gil und Tatiana Romanova (März 2004). „ϕ-functions for complex 2D-objects“. In: *4OR* 2, S. 69–84. DOI: 10.1007/s10288-003-0027-1.

[Suhl und Mellouli, 2007]
Leena Suhl und Taieb Mellouli (2007). *Optimierungssysteme: Modelle, Verfahren, Software, Anwendungen (Springer-Lehrbuch)*. Berlin, Heidelberg: Springer-Verlag. ISBN: 35402-61192.

[Szykman und Cagan, 1997]
S. Szykman und Jonathan Cagan (März 1997). „Constrained Three-Dimensional Component Layout Using Simulated Annealing“. In: *Journal of Mechanical Design* 119.1, S. 28–35. ISSN: 1050-0472. URL: https://doi.org/10.1115/1.2828785.

[Takahama und Sakai, 2006]
Tetsuyuki Takahama und Setsuko Sakai (Jan. 2006). „Constrained Optimization by the ϵ Constrained Differential Evolution with Gradient-Based Mutation and Feasible Elites“. In: *2006 IEEE Congress on Evolutionary Computation, CEC 2006*, S. 1–8. DOI: 10.1109/CEC.2006.1688283.

[Tawarmalani und Sahinidis, 2002]
Mohit Tawarmalani und Nikolaos Sahinidis (Jan. 2002). *Convexification and global optimization in continuous and mixed-integer nonlinear programming. Theory, algorithms, software, and applications*. Bd. 65. ISBN: 978-1-4419-5235-6. DOI: 10.1007/978-1-4757-3532-1.

[Teng et al., 2001]
Hong-fei Teng, Shou-lin Sun, De-quan Liu und Yan-zhao Li (2001). „Layout optimization for the objects located within a rotating vessel — a three-dimensional packing problem with behavioral constraints“. In: *Computers & Operations Research* 28.6, S. 521–535. ISSN: 0305-0548. DOI: https://doi.org/10.1016/S0305-0548(99)00132-X.

[The Apache Software Foundation, 2022]
The Apache Software Foundation (2022). *Apache Commons*. URL: https://commons.apache.org/.

[Tiwari et al., 2014]
Santosh Tiwari, Hong Dong, Georges Fadel, Peter Fenyes und Artemis Kloess (Juli 2014). „A physically-based shape morphing algorithm for packing and layout applications“. In: *International Journal on Interactive Design and Manufacturing (IJIDeM)* 9. DOI: 10.1007/s12008-014-0237-0.

[Tiwari et al., 2008]
Santosh Tiwari, Georges Fadel und Peter Fenyes (Aug. 2008). „A Fast and Efficient Compact Packing Algorithm for Free-Form Objects“. In: International Design Engineering Technical Conferences and Computers and Information in Engineering Conference Volume 1: 34th Design Automation Conference, Parts A and B, S. 543–552. DOI: 10.1115/DETC2008-50097. URL: https://doi.org/10.1115/DETC2008-50097.

[Tiwari et al., 2010]
Santosh Tiwari, Georges Fadel und Peter Fenyes (Juni 2010). „A Fast and Efficient Compact Packing Algorithm for SAE and ISO Luggage Packing Problems“. In: *Journal of Computing and Information Science in Engineering* 10.2. 021010. ISSN: 1530-9827. DOI: 10.1115/1.3330440. URL: https://doi.org/10.1115/1.3330440.

[Vanderbei, 2013]
Robert J. Vanderbei (2013). *Linear programming: foundations and extensions*. New York: Springer. ISBN: 9781461476290.

[Vielma, 2015]
Juan Vielma (2015). „Mixed Integer Linear Programming Formulation Techniques“. In: *SIAM Review* 57.1, S. 3–57. DOI: 10.1137/130915303.

[Vielma und Nemhauser, 2008]
Juan Vielma und George Nemhauser (Mai 2008). „Modeling Disjunctive Constraints with a Logarithmic Number of Binary Variables and Constraints“. In: *Mathematical Programming* 128, S. 199–213. DOI: 10.1007/978-3-540-68891-4_14.

[Vielma und Nemhauser, 2011]
Juan Vielma und George Nemhauser (2011). „Modeling disjunctive constraints with a logarithmic number of binary variables and constraints“. In: *Mathematical Programming* 128, S. 49–72.

[Wäscher et al., 2007]
Gerhard Wäscher, Heike Haußner und Holger Schumann (2007). „An improved typology of cutting and packing problems“. In: *European journal of operational research* 183.3, S. 1109–1130.

[Weismantel, 2003]
Robert Weismantel (2003). „Diskrete Optimierung“. In: *Magdeburger Wissenschaftsjournal online* 1-2, S. 17–24.

[Weller und Zachmann, 2008]
René Weller und Gabriel Zachmann (2008). „Inner Sphere Trees and Their Application to Collision Detection“. In: *Virtual Realities*.

[Welzl, 1991]
Emo Welzl (1991). „Smallest enclosing disks (balls and ellipsoids)“. In: *New Results and New Trends in Computer Science*. Hrsg. von Hermann Maurer, S. 359–370.

[Wu et al., 2017]
Hongteng Wu, Stephen Leung, Yain-whar Si, Defu Zhang und Adi Lin (2017). „Three-stage heuristic algorithm for three-dimensional irregular packing problem“. In: *Applied Mathematical Modelling* 41, S. 431–444. ISSN: 0307-904X. DOI: https://doi.org/10.1016/j.apm.2016.09.018.

[Yakovlev und Kartashov, 2018]
Sergiy Yakovlev und Oleksii Kartashov (2018). „System Analysis and Classification of Spatial Configurations“. In: *2018 IEEE First International Conference on System Analysis & Intelligent Computing (SAIC)*, S. 1–4.

[Yakovlev et al., 2018]
Sergiy Yakovlev, Oleksii Kartashov und Kyryl Korobchynskyi (2018). „The Informational Analytical Technologies of Synthesis of Optimal Spatial Configuration“. In: *2018 IEEE 13th International Scientific and Technical Conference on Computer Sciences and Information Technologies (CSIT)* 2, S. 140–143. DOI: 10.1109/STC-CSIT.2018.8526704.

[Yi et al., 2008]
Miao Yi, Georges Fadel und Vladimir Gantovnik (Jan. 2008). „Vehicle configuration design with a packing genetic algorithm“. In: *International Journal of Heavy Vehicle Systems* 15. DOI: 10.1504/IJHVS.2008.022252.

[Yin und Cagan, 2000]
Su Yin und Jonathan Cagan (Jan. 2000). „An Extended Pattern Search Algorithm for Three-Dimensional Component Layout“. In: *Journal of Mechanical Design* 122.1, S. 102–108. ISSN: 1050-0472. DOI: 10.1115/1.533550. eprint: https://asmedigitalcollection.asme.org/mechanicaldesign/article-pdf/122/1/102/5796987/102_1.pdf. URL: https://doi.org/10.1115/1.533550.

[Yin und Cagan, 2004]
Su Yin und Jonathan Cagan (Jan. 2004). „Exploring the Effectiveness of Various Patterns in an Extended Pattern Search Layout Algorithm“. In: *Journal of Mechanical Design* 126. DOI: 10.1115/1.1641185.

[Yin et al., 2004]
Su Yin, Jonathan Cagan und Peter Hodges (März 2004). „Layout Optimization of Shapeable Components With Extended Pattern Search Applied to Transmission Design“. In: *Journal of Mechanical Design* 126.1, S. 188–191. ISSN: 1050-0472. DOI: 10.1115/1.1637663. URL: https://doi.org/10.1115/1.1637663.

[Zawidzki und Szklarski, 2020]
Machi Zawidzki und Jacek Szklarski (2020). „Multi-objective optimization of the floor plan of a single story family house considering position and orientation“. In: *Advances in Engineering Software* 141, S. 102766. ISSN: 0965-9978. DOI: https://doi.org/10.1016/j.advengsoft.2019.102766. URL: https://www.sciencedirect.com/science/article/pii/S0965997819301310.

[D. Zhang et al., 2012]
Defu Zhang, Yu Peng und Stephen Leung (2012). „A heuristic block-loading algorithm based on multi-layer search for the container loading problem“. In: *Computers & Operations Research* 39.10, S. 2267–2276. ISSN: 0305-0548. DOI: https://doi.org/10.1016/j.cor.2011.10.019.

[H. Zhang, 2011]
Hong Zhang (Mai 2011). „Multiple Particle Swarm Optimizers with Inertia Weight wi-

th Diversive Curiosity and Its Performance Test“. In: *IAENG International Journal of Computer Science* 38, S. 134–145.

[Zheng und Ren, 2020]
Hao Zheng und Yue Ren (Aug. 2020). „Architectural Layout Design through Simulated Annealing Algorithm“. In.

[Zhi-Guo und Hong-Fei, 2003]
Sun Zhi-Guo und Teng Hong-Fei (2003). „Optimal layout design of a satellite module“. In: *Engineering Optimization* 35.5, S. 513–529. URL: https://doi.org/10.1080/0305215031000160233 5.

[Zhu et al., 2012]
Jihong Zhu, Weihong Zhang, Liang Xia, Qiao Zhang und David Bassir (2012). „Optimal packing configuration design with finite-circle method“. In: *Journal of Intelligent & Robotic Systems* 67.3-4, S. 185–199.

Abbildungsverzeichnis

Tabellenverzeichnis