MARTIN-LUTHER-UNIVERSITÄT
HALLE-WITTENBERG
Juristische & Wirtschaftswissenschaftliche
Fakultät
Wirtschaftswissenschaftlicher Bereich
Lehrstuhl für Wirtschaftsinformatik &
Operations Research
Prof. Dr. Taïeb Mellouli

Masterarbeit

ENTWICKLUNG EINES KI-BASIERTEN ANSATZES ZUR GENERIERUNG MATHEMATISCHER MODELLE UND METHODIKEN FÜR PRAKTISCHE OPTIMIERUNGSAUFGABEN

vorgelegt von

Wienke Jensen

217233660

22.07.2025

Abstract

Die vorliegende Masterarbeit befasst sich mit der Entwicklung eines KI-basierten Ansatzes zur automatisierten Generierung mathematischer Modelle für praktische Optimierungsaufgaben. Ziel ist es, mit Hilfe moderner Large Language Models wie GPT-4o natürlichsprachliche Problemstellungen aus verschiedenen Anwendungsbereichen in formal korrekte, algebraische Optimierungsmodelle, insbesondere in der Sprache AMPL, zu übersetzen. Dazu wird ein prototypisches System entwickelt, das relevante Informationen aus unstrukturierten Texten extrahiert, strukturiert und in mathematische Modelle überführt. Die Leistungsfähigkeit des Systems wird anhand realitätsnaher Fallstudien aus Bereichen wie Logistik, Produktion und Gesundheitswesen evaluiert. Die Arbeit liefert einen Beitrag zur Demokratisierung mathematischer Optimierung und zeigt sowohl Potenziale als auch Grenzen KI-basierter Modellgenerierung im betrieblichen Kontext auf. Abschließend werden die Ergebnisse kritisch reflektiert und ein Ausblick auf zukünftige Forschungs- und Entwicklungsfelder gegeben.

Inhaltsverzeichnis

1. Einleitung

Die fortschreitende Digitalisierung und die wachsende Verfügbarkeit großer Datenmengen beeinflussen die Wirtschaft und die Wissenschaft gleichermaßen und verändern die Art und Weise, wie Entscheidungsprobleme angegangen und gelöst werden.[1] Im Zuge dieser Entwicklung hat sich die mathematische Optimierung in den letzten Jahrzehnten als Schlüsseltechnologie für unterschiedlichste Anwendungsfelder etabliert. Sie ermöglicht es, komplexe Entscheidungen unter zahlreichen Rahmenbedingungen effizient und systematisch zu treffen und ist heute aus Branchen wie Logistik, Produktion, Energieversorgung oder Gesundheitswesen nicht mehr wegzudenken. Ein wesentlicher Treiber dieser Entwicklung sind die enormen Fortschritte in der algorithmischen Entwicklung und der Leistungsfähigkeit moderner Optimierungssolver. Wie das Fraunhofer-Institut für Integrierte Schaltungen (Fraunhofer SCS) betont, ist es heute möglich, lineare Optimierungsprobleme mit Milliarden von Variablen innerhalb kürzester Zeit zu lösen, während auch anspruchsvolle ganzzahlige Probleme mittlerweile im Millionenbereich bearbeitet werden können. Diese beeindruckenden Leistungssteigerungen beruhen nicht nur auf der kontinuierlichen Weiterentwicklung der Computerhardware, sondern insbesondere auf methodisch-algorithmischen Innovationen, die in den vergangenen Jahren zu Effizienzsteigerungen um den Faktor von mehreren Millionen geführt haben. Damit eröffnet die mathematische Optimierung in der Praxis neue Möglichkeiten, komplexe Systeme nahezu in Echtzeit zu steuern und auf wechselnde Rahmenbedingungen flexibel zu reagieren.[2]

Parallel zu diesen Entwicklungen hat das Aufkommen leistungsfähiger Methoden der Künstlichen Intelligenz, insbesondere der sogenannten Large Language Models wie GPT-4o (Generative Pre-trained Transformer), die Perspektive auf die mathematische Optimierung grundlegend verändert. Diese modernen Sprachmodelle, die auf umfangreichen Text- und Datenkorpora trainiert werden, sind heute in der Lage, natürliche Sprache nicht nur zu verstehen und zu analysieren, sondern auch eigenständig strukturierte Texte und sogar Programmcode zu generieren. Damit eröffnen sich ganz neue Ansätze, um Optimierungsprobleme zu formulieren, zu lösen und die Schnittstelle zwischen Mensch und mathematischer Modellierung grundlegend zu vereinfachen.[3]

In Wissenschaft und Praxis wächst das Interesse an der automatisierten Nutzung von KI-Sprachmodellen, da sich dadurch erhebliche Potenziale für effizientere, flexiblere und weniger fehleranfällige Arbeitsprozesse ergeben, wie etwa bei der automatischen

[1] Vgl. Biermann-Wehmeyer (2024).
[2] Vgl. Fraunhofer SCS (o.J.).
[3] Vgl. Kelbert et al. (2023).

Generierung von Texten oder Programmcodes.[4] Viele Optimierungsprobleme aus der Praxis werden bislang nicht in Form mathematischer Modelle umgesetzt, da häufig das erforderliche Fachwissen, die nötigen Ressourcen oder die entsprechende Aufmerksamkeit innerhalb der Organisation fehlen.[5]

Indem KI-basierte Systeme den Schritt von der Problemformulierung zur mathematischen Modellierung weitgehend automatisieren, wird der Einsatz moderner Optimierungsverfahren in Unternehmen und Organisationen potenziell erheblich erleichtert. Gleichzeitig ergeben sich neue wissenschaftliche Fragestellungen, etwa hinsichtlich der Zuverlässigkeit, Belastbarkeit und Nachvollziehbarkeit solcher automatisiert generierten Modelle sowie der Grenzen, die aktuellen KI-Systemen bei komplexen oder mehrdeutigen Aufgaben noch gesetzt sind.[6]

Vor diesem Hintergrund steht im Mittelpunkt der vorliegenden Arbeit die Frage, inwieweit es mit Hilfe eines KI-basierten Systems auf Basis moderner Large Language Models möglich ist, mathematische Optimierungsmodelle automatisiert aus natürlichsprachlichen Problemstellungen zu generieren und welche technischen, praktischen und qualitativen Anforderungen sowie Grenzen sich bei der Entwicklung und Anwendung eines solchen Ansatzes in praxisorientierten Anwendungsfällen ergeben. Dabei werden sowohl die technische Umsetzbarkeit und Leistungsfähigkeit eines solchen Systems als auch der praktische Nutzen und die bestehenden Limitationen im Anwendungskontext betrachtet.

Vor diesem Hintergrund verfolgt die vorliegende Arbeit das Ziel, ein System zu entwickeln, das mit Hilfe moderner KI-Technologie, insbesondere eines Large Language Models, in der Lage ist, natürlichsprachliche Optimierungsprobleme automatisiert in formal korrekte mathematische Modelle zu überführen. Die generierten Modelle sollen dabei in einer etablierten algebraischen Modellierungssprache, im vorliegenden Fall AMPL, abgebildet werden, um eine breite Anwendbarkeit und Kompatibilität mit bestehenden Optimierungsverfahren zu gewährleisten. Darüber hinaus wird das entwickelte System im Rahmen von praxisorientierten Fallstudien evaluiert, um zu prüfen, inwiefern die automatisierte Modellgenerierung tatsächlich zur Lösung realer betrieblicher Problemstellungen beitragen kann.

Der thematische Schwerpunkt dieser Arbeit liegt damit im Schnittfeld zwischen Operations Research, Wirtschaftsinformatik und moderner KI-Forschung. Die grundlegende Motivation besteht darin, einen Beitrag zur Weiterentwicklung intelligenter Werkzeuge für die mathematische Optimierung zu leisten und zu untersuchen, wie die Potenziale generativer Sprachmodelle für die Demokratisierung und Effizienzsteigerung im

[4] Vgl. Kelbert et al. (2023).
[5] Vgl. Sudermann-Merx (2023), S. 195 f.
[6] Vgl. Kelbert et al. (2023).

Modellierungsprozess nutzbar gemacht werden können. Das Thema ist eingebettet in die aktuelle wissenschaftliche Diskussion um die Rolle Künstlicher Intelligenz in der Wissensarbeit, die Transformation von Arbeitsprozessen und die zunehmende Automatisierung analytischer Aufgaben in Unternehmen und Organisationen.

Im Mittelpunkt der vorliegenden Untersuchung steht die Entwicklung und Evaluation eines prototypischen Systems, das in der Lage ist, Optimierungsprobleme aus natürlichsprachlichen Beschreibungen automatisiert in mathematische Modelle zu übersetzen und diese mit Hilfe moderner Optimierungsverfahren zu lösen. Hierzu werden zentrale Begriffe wie Large Language Models, mathematische Optimierungsmodelle und die Modellierungssprache AMPL im Kontext dieser Arbeit definiert. Der Fokus liegt auf klassischen, deterministischen Optimierungsproblemen, insbesondere auf linearen, ganzzahligen und gemischt-ganzzahligen Modellen, wie sie in der betrieblichen Praxis häufig vorkommen. Nichtlinearere, stochastische oder dynamische Optimierungsmodelle werden ausdrücklich ausgeklammert, da sie zusätzliche methodische und technische Herausforderungen mit sich bringen, die im Rahmen dieser Arbeit nicht adressiert werden.

Die Arbeit gliedert sich in mehrere inhaltliche Abschnitte. Nach dieser Einleitung werden zunächst die theoretischen und methodischen Grundlagen erläutert, um das Verständnis für die Funktionsweise von KI-basierten Modellgenerierungssystemen zu schaffen. Dies umfasst sowohl die zentralen Prinzipien der Künstlichen Intelligenz und neuronaler Netze als auch die Struktur und Methodik mathematischer Optimierungsmodelle. Daran anschließend erfolgt eine detaillierte Darstellung des Forschungsstands, wobei aktuelle Ansätze zur KI-gestützten Modellgenerierung ebenso beleuchtet werden wie einschlägige Forschungsarbeiten aus dem Bereich Optimierung und KI. Darüber hinaus werden bestehende Modellierungsframeworks diskutiert und hinsichtlich ihrer jeweiligen Stärken und Schwächen kritisch gewürdigt.

Im empirisch-methodischen Teil der Arbeit werden zunächst das Forschungsdesign, die methodische Vorgehensweise und die Anforderungen an das entwickelte System erläutert. Es wird gezeigt, wie auf Basis einer iterativen und modularen Entwicklung ein Prototyp entsteht, der den gesamten Prozess von der Problemaufnahme über die automatisierte Modellgenerierung bis zur rechnergestützten Lösung abbildet. Besonderes Augenmerk gilt dabei der Trennung von Modellstruktur und Instanzdaten, der Gestaltung benutzerfreundlicher Schnittstellen und der Integration leistungsfähiger Solver.

Die eigentliche Entwicklung des Systems bildet den Kern der Arbeit. Hier werden sowohl die Architektur als auch die Funktionsweise der einzelnen Komponenten und Module detailliert beschrieben, angefangen bei der Nutzereingabe bis hin zur Ergebnispräsentation und Validierung. Im Anschluss daran wird das System anhand ausgewählter praxisnaher

Fallstudien evaluiert, um die Leistungsfähigkeit und Grenzen des entwickelten Ansatzes in realitätsnahen Szenarien zu demonstrieren und kritisch zu reflektieren.

Den Abschluss der Arbeit bildet eine umfassende Zusammenfassung der zentralen Ergebnisse, die Beantwortung der Forschungsfrage sowie ein Ausblick auf weiterführende Forschungsfragen und Entwicklungsperspektiven, die sich aus der Auseinandersetzung mit dem Thema ergeben.

2. Theoretische und methodische Grundlagen

2.1 Künstliche Intelligenz und Sprachverarbeitung

2.1.1 Sprachmodelle zur strukturierten Wissensverarbeitung

Ein zentraler Forschungstrend im Bereich der natürlichen Sprachverarbeitung ist die sogenannte knowledge-enhanced text generation, also die wissensgestützte Textgenerierung. Dabei wird zwischen internem und externem Wissen unterschieden: Internes Wissen bezieht sich auf Informationen, die direkt aus dem gegebenen Eingabetext abgeleitet werden können, beispielsweise Themen, Schlüsselbegriffe, linguistische Merkmale oder interne Textgraphstrukturen. Externes Wissen hingegen stammt aus außerhalb der Eingabe liegenden Quellen wie Wissensdatenbanken, externen Wissensgraphen oder begleitenden Hintergrunddokumenten. Beide Wissensarten können über geeignete Repräsentationsverfahren in den Generierungsprozess integriert werden, um so nicht nur die Kohärenz, sondern auch die inhaltliche Fundierung und semantische Interpretierbarkeit der erzeugten Texte zu verbessern. Die gezielte Kombination solcher Wissensquellen dient dabei nicht nur der Inhaltsanreicherung, sondern auch der Strukturierung und expliziten Verankerung semantischer Konzepte innerhalb der generierten Sprache. Diese systematische Einbindung domänenrelevanter Informationen bildet eine konzeptionelle Grundlage für die strukturierte Nutzung und Verarbeitung von Fachwissen durch Large Language Models (LLMs) im Kontext dieser Arbeit.[7]

Sprachmodelle wurden ursprünglich entwickelt, um auf Grundlage vorhergehender Textelemente die wahrscheinlichste Fortsetzung zu bestimmen. Ihre Anwendung in Bereichen wie der automatischen Spracherkennung oder der maschinellen Übersetzung verdeutlicht, dass bereits einfache Vorhersagemodelle leistungsfähige Ergebnisse liefern können[8]. Die Fähigkeit dieser Modelle, sich mit zunehmender Datenmenge und Rechenkapazität systematisch zu skalieren, bildet die konzeptionelle Brücke zu den heute verwendete LLMs.

In der aktuellen Forschung bezeichnet man mit LLMs eine Klasse tiefenlernender Modelle, die darauf ausgerichtet sind, sprachliche Zusammenhänge zu erfassen, indem sie auf Grundlage großer Textkorpora die Wahrscheinlichkeiten sprachlicher Folgeelemente berechnen. Der zugrunde liegende Prozess, das sogenannte Language Modeling, zielt darauf ab, aus einem gegebenen Textkontext das nächste oder ein fehlendes Wort möglichst plausibel zu ergänzen[9]. Anders als klassische Modelle generieren LLMs nicht nur

[7] Vgl. Yu et al. (2022), S. 3.
[8] Vgl. Hestness et al. (2017), S. 1.
[9] Vgl. Blank (2023), S. 987.

syntaktisch korrekte Sätze, sondern identifizieren auch statistische Regelmäßigkeiten und semantische Strukturen in der Sprache. Damit ermöglichen sie die automatische Erstellung kohärenter Texte mit beachtlicher Generalisierungsfähigkeit. Systeme wie GPT-3 oder GPT-4 gelten in diesem Zusammenhang als leistungsfähigste Implementierungen maschineller Sprachverarbeitung.

Dabei stellt sich jedoch die grundlegende Frage, ob LLMs über die rein technische Fähigkeit der Vorhersage hinaus auch ein kognitives Modell von Sprache entwickeln. Blank (2023) argumentiert, dass linguistische Vorhersage zwar auch in der menschlichen Sprachverarbeitung ein zentraler Prozess sei, maschinelle Prädiktion aber nicht mit menschlichem Sprachverstehen gleichgesetzt werden könne[10]. Zwar arbeiten LLMs mit strukturellen Repräsentationen, wie syntaktischen Abhängigkeiten, thematischen Rollen oder argumentativen Funktionen, doch entsteht daraus kein mentales Weltmodell im engeren Sinne. Vielmehr bleibt die interne Repräsentation sprachlicher Inhalte auf statistische Assoziationen beschränkt.

Ein weiterer zentraler Aspekt bezieht sich auf die Trainingsgrundlage moderner LLMs. Diese Modelle werden auf extrem umfangreichen, meist öffentlich verfügbaren Textkorpora trainiert, etwa aus Wikipedia, digitalisierten Büchern oder groß angelegten Web-Datensätzen. Aufgrund dieser Datenbasis gelten sie als kollektive Spiegel menschlicher Sprach- und Wissensfragmente[11]. Auch ohne domänenspezifisches Finetuning, erzielen sie Ergebnisse in Bereichen wie Textklassifikation, Dialogverarbeitung oder der automatischen Modellierung von Optimierungsaufgaben. Gleichzeitig wird kritisch angemerkt, dass LLMs unter Umständen lediglich trainierte Muster reproduzieren, anstatt echtes Verständnis zu entwickeln.

Ein zentrales Mittel zur Steuerung der Ausgabe von LLMs ist das sogenannte Prompting. Es ermöglicht, den Modellen Aufgabenstellungen in spezifischer Form zu präsentieren, sodass sie die gewünschte Art der Antwort generieren. Besonders etabliert hat sich das Few-Shot Prompting, bei welchem dem Modell einige Beispiele für die zu lösende Aufgabe als Kontext übergeben werden, etwa in Form von Satzpaaren oder Problem-Lösungs-Beispielen. Während der Inferenz bleiben die Modellgewichte dabei unverändert; das Modell zieht seine Schlussfolgerungen ausschließlich aus dem gegebenen Kontext[12]. Diese Methode reduziert den Bedarf an umfangreichen, aufgabenspezifischen Trainingsdaten und erlaubt es dem Modell, auch auf neue Aufgaben zu reagieren, für die es nicht explizit trainiert wurde. Aufbauend darauf wurden weitere Prompting-Strategien entwickelt, etwa das Chain-of-Thought Prompting, das durch das Einfügen logischer Zwischenschritte die

[10] Vgl. Blank (2023), S. 987.
[11] Vgl. Wulff et al. (2025), S. 19.
[12] Vgl. Brown et al. (2020), S. 1879.

Problemlösungsfähigkeit verbessert, oder Self-Consistency- und Tree-of-Thought-Ansätze, die mehrere Lösungspfade erzeugen und auf Konsistenz prüfen. Diese Techniken zeigen, dass die Leistungsfähigkeit von LLMs nicht nur von ihrer Architektur, sondern wesentlich von der Art der Interaktion über Prompts abhängt.[13]

2.1.2 Technologische Grundlagen neuronaler Netze

Künstliche neuronale Netze (KNN) stellen eine zentrale Methode im Bereich des maschinellen Lernens dar, die auf der strukturellen und funktionalen Nachbildung biologischer Nervensysteme basiert. Dabei bestehen künstliche Neuronen aus simplen Verarbeitungseinheiten, die über gewichtete Verbindungen organisiert sind. Durch die Verknüpfung mehrerer dieser Einheiten in Schichten entstehen Netzwerke, die komplexe Aufgaben wie Klassifikation, Mustererkennung oder Prognose erlernen können. Die Architektur eines neuronalen Netzes gliedert sich typischerweise in eine Eingabeschicht, mehrere versteckte Schichten (Hidden Layers) und eine Ausgabeschicht. Die Neuronen jeder Schicht sind vollständig mit den Neuronen der nächsten Schicht verbunden. Die Neuronen sind dabei über gewichtete Verbindungen miteinander verknüpft. Durch Aktivierungsfunktionen werden Eingaben in Zwischenergebnisse transformiert, die durch das Netzwerk propagiert werden.[14]

Im Lernprozess neuronaler Netze besteht das Ziel darin, die Gewichtungen innerhalb des Netzes so zu optimieren, dass es bestimmte Aufgaben, wie etwa Sprachverarbeitung oder die Zuordnung von Eingaben zu vordefinierten Klassen (Klassifikation), erfolgreich ausführen kann. Dies geschieht durch den Einsatz des Backpropagation-Verfahrens, bei dem der Fehler zwischen der vorhergesagten und der tatsächlichen Ausgabe über die Schichten des Netzes hinweg rückwärts propagiert wird. Dabei werden die Gewichte iterativ so angepasst, dass der Fehler minimiert wird. Dieser Prozess erlaubt es dem Modell, auch bei komplexen, mehrschichtigen Architekturen relevante Verantwortlichkeiten für Fehler korrekt zuzuweisen, ein Prinzip, das als Credit Assignment bezeichnet wird und zentral für das sogenannte tiefe Lernen (Deep Learning) ist.[15]

Darüber hinaus verfügen künstliche neuronale Netze über bemerkenswerte Anpassungs- und Generalisierungsfähigkeit: Sie sind in der Lage, auch auf unbekannte Eingaben zu reagieren und implizite Muster zu erkennen, ohne dass deterministische Regeln vorgegeben werden müssen.[16]

[13] Vgl. Wulff et al. (2025), S. 19.
[14] Vgl. Shu (2020), S. 191ff.
[15] Vgl. Schmidhuber (2015), S. 86f.
[16] Vgl. Shu (2020), S. 196.

Buscema (1998) begründet seine theoretische Herangehensweise, indem er auf der Mikroebene die semantischen Grundbausteine von Neuronen und Verbindungen (sogenannte Nodes und Connections) untersucht. Er beschreibt die verschiedenen Verbindungstypen (symmetrisch, antisymmetrisch, bidirektional, monodirektional, reflexiv) und klassifiziert Netze nach ihrer Struktur und Dynamik.[17] Besonders relevant ist dabei die Unterscheidung zwischen Netzwerken, die durch Input-Daten überwacht werden (Monitored), sowie selbstorganisierenden Netzwerken (AutoPoietic), abhängig davon, ob und in welcher Form Zielwerte (Targets) vorgegeben sind.[18]

In seiner späteren Übersicht betont Buscema (2002), dass neuronale Netze besonders in Szenarien effektiv sind, in denen keine expliziten Regeln zur Verfügung stehen, aber große Mengen unstrukturierter oder komplexer Daten vorhanden sind.[19] In solchen Fällen sind KNNs in der Lage, durch Trainingsprozesse sogenannte fuzzy rules[20] zu extrahieren, die eine Annäherung an zugrundeliegende Beziehungen ermöglichen.[21] Dies unterscheidet sie von klassischen regelbasierten Systemen, die auf fest kodierten Regeln beruhen.

Insgesamt zeigt sich, dass neuronale Netze flexible Werkzeuge darstellen, die sich sowohl zur Lösung strukturierter Klassifikationsprobleme als auch für explorative Analyse, und Prognoseverfahren eignen, insbesondere in Anwendungsbereichen, in denen klassische mathematische Modelle an ihre Grenzen stoßen.

2.2 Mathematische Modellierung in der Optimierung

2.2.1 Struktur und Elemente formaler Optimierungsmodelle

Mathematische Optimierungsmodelle dienen der formalen Abbildung realer Entscheidungsprobleme, bei denen eine bestmögliche Lösung unter Einhaltung vorgegebener Bedingungen gefunden werden soll. Die Grundstruktur solcher Modelle besteht stets aus drei Hauptelementen: Entscheidungsvariablen, Zielfunktion und Nebenbedingungen. Die Entscheidungsvariablen repräsentieren dabei jene Größen, deren Werte im Rahmen des Optimierungsprozesses festgelegt werden. Die Zielfunktion beschreibt mathematisch das angestrebte Ziel, wie die Minimierung von Kosten oder die Maximierung eines Gewinns, und wird als Funktion der Entscheidungsvariablen formuliert. Die Nebenbedingungen legen

[17] Vgl. Buscema (1998), S. 18f.
[18] Vgl. Ebd., S. 33.
[19] Vgl. Buscema (2002), S. 1096.
[20] Regeln, die Zugehörigkeiten oder Zusammenhänge nicht exakt, sondern mit graduellen Abstufungen abbilden
[21] Vgl. Buscema (2002), S. 1097.

zulässige Wertebereiche und weitere Restriktionen fest, um sicherzustellen, dass nur realistisch umsetzbare Lösungen berücksichtigt werden.[22]

Der Begriff mathematical programming bezeichnet dabei die mathematische Planung und ist von computer programming zu unterscheiden. Mathematical programming steht für die Entwicklung formaler Modelle zur Optimierung praktischer Entscheidungen, unabhängig vom Einsatz digitaler Rechentechnik.[23] Besonders verbreitet sind lineare Optimierungsmodelle (Lineare Programme), bei denen sowohl die Zielfunktion als auch sämtliche Nebenbedingungen linear formuliert sind.

Formal wird die Zielfunktion in der allgemeinen Form dargestellt als:

$$z = F(x_1, \dots, x_n) = \sum_{j=1}^{n} c_j x_j + d,$$

wobei x_j die Entscheidungsvariablen, c_j die Koeffizienten und d eine Konstante ist. In praktischen Anwendungen kann der Summand d in der Zielfunktion weggelassen werden, da er auf das Optimierungsergebnis keinen Einfluss hat.

Das Ziel besteht darin, den Funktionswert z unter Einhaltung einer Menge von linearen Nebenbedingungen zu optimieren. Diese Nebenbedingungen beschränken die zulässigen Werte der Entscheidungsvariablen und können in drei Formen auftreten:

$$\sum_{j=1}^{n} a_{ij} x_j \leq b_i \,, \qquad \sum_{j=1}^{n} a_{ij} x_j = b_i \,, \qquad \sum_{j=1}^{n} a_{ij} x_j \geq b_i \,,$$

für $i = 1, \dots, m$. Hierbei sind a_{ij} die Strukturkoeffizienten und b_i gibt den Grenzwert der jeweiligen Restriktion an.

Zusätzlich treten in vielen praktischen Problemstellungen sogenannte Nichtnegativitätsbedingungen auf, die fordern, dass einzelne oder alle Entscheidungsvariablen nicht negativ sein dürfen:

$x_j \geq 0$ für $j = 1, \dots, n$.

Wesentlich bei der Modellbildung ist, dass alle auftretenden Beziehungen linear sind. Diese Linearität betrifft sowohl die Zielfunktion als auch sämtliche Nebenbedingungen. Sie ist nicht nur entscheidend für die mathematische Struktur, sondern auch für die Auswahl effizienter Lösungsverfahren und die Anwendbarkeit automatisierter Modellgenerierung.[24]

[22] Vgl. Wang, Zhao (2023), S. 87.
[23] Vgl. Williams (2013), S5.
[24] Vgl. Koop, Moock (2023), S. 41f.

2.2.2 Modellbildung in AMPL

Die Modellierung mathematischer Optimierungsprobleme erfolgt in der Praxis häufig mit sogenannten algebraischen Modellierungssprachen. Eine etablierte Sprache in diesem Bereich ist AMPL (A Mathematical Programming Language), die speziell für die Formulierung, Strukturierung und Analyse von Optimierungsmodellen konzipiert wurde. AMPL eignet sich insbesondere für Modelle, die mit endlich vielen Variablen, Gleichungen, Ungleichungen und Zielfunktionen operieren und sich durch algebraische Ausdrücke beschreiben lassen. Ein zentrales Gestaltungsprinzip von AMPL ist die möglichst intuitive Übertragung mathematischer Modelle in eine computerlesbare Form. Die Sprache orientiert sich bewusst an der gewohnten algebraischen Notation und erlaubt eine deklarative Beschreibung der Modellstruktur. Modelle lassen sich dabei zeilenweise von oben nach unten lesen, ohne dass spätere Definitionen vorweggenommen werden müssen.

Ein AMPL-Modell gliedert sich typischerweise in drei Komponenten:

- Deklarationsteil zur Definition von Mengen, Parametern, Variablen und Zielfunktionen
- Datenteil zur Zuweisung konkreter Werte
- Kommandoteil zur Steuerung der Lösung und Auswertung

Die Syntax orientiert sich an bekannten Programmiersprachen wie C oder awk und verwendet beispielsweise eckige Klammern zur Indizierung.

Ein einfaches Beispiel für ein lineares Minimierungsmodell lautet:

```
set S;

var x{S} >= 0;

param p{S};

minimize Cost: sum{i in S} p[i]*x[i];
```

Hier steht die Menge S für den Indexbereich der Entscheidungsvariablen x und Parameter p. Die Zielfunktion Cost minimiert die gewichtete Summe der Entscheidungsvariablen und entspricht der algebraischen Darstellung

$$\sum_{i \in S} p_i x_i$$

Ein wesentliches Merkmal von AMPL ist die strikte Trennung zwischen Modellstruktur und Instanzdaten. Dadurch lassen sich Modelle flexibel wiederverwenden und effizient auf unterschiedliche Datensätze anwenden.[25]

Die eigentliche Lösung erfolgt nicht innerhalb von AMPL, sondern durch externe Solver. Über eine standardisierte Schnittstelle werden vollständige Probleminstanzen an spezialisierte Optimierungsverfahren übergeben. Vorab können automatische Vereinfachungen vorgenommen werden (Presolve), und nur tatsächlich benötigte Modellbestandteile werden instanziiert oder aktualisiert (Lazy Evaluation), was insbesondere bei großen Modellen erhebliche Vorteile bringt.[26]

AMPL verringert darüber hinaus den Bedarf an manueller Programmierung. Modelle können in der Denkweise des Modellierers formuliert werden, ohne die Zwischenschritte algorithmischer Umsetzung, etwa bei der Erstellung von Koeffizientenmatrizen. Dies beschleunigt den Entwicklungsprozess und reduziert Fehlerquellen. Dank seiner algebraischen Struktur eignet sich AMPL für eine breite Palette mathematischer Programme, von linearen und nichtlinearen bis zu ganzzahligen Modellen. Darüber hinaus erlaubt die Sprache die Darstellung erweiterter Konzepte wie Netzwerkflüsse, stückweise lineare Beziehungen sowie den Import von Tabellen aus externen Datenquellen.[27]

2.3 Optimierungsverfahren

2.3.1 Lineare Programmierung

Optimierungsverfahren stellen die methodische Grundlage dar, um die in mathematischen Modellen formulierten Entscheidungsprobleme rechnerisch zu lösen. Während Kapitel 2.2 die strukturelle Modellierung in Form von Zielfunktionen, Nebenbedingungen und Entscheidungsvariablen behandelt hat, rückt nun die rechnerische Lösung dieser Modelle in den Fokus. Abhängig von der Struktur des Modells, insbesondere der Eigenschaften der Variablen und der Funktionalität der Restriktionen, kommen unterschiedliche Optimierungsmethoden zum Einsatz.

Die lineare Programmierung befasst sich mit der Beschreibung eines Problems anhand eines mathematischen Modells, das ausschließlich lineare Funktionen enthält. Die Voraussetzung für ein lineares Programmierungsproblem ist, dass sowohl die Zielsetzung als auch die mathematischen Funktionen des Modells in linearer Form vorliegen. Dabei ist der Begriff „Programmierung“ nicht im Sinne der Softwareentwicklung zu verstehen, sondern als

[25] Vgl. Gay (2015), S. 95ff.
[26] Vgl. Gay (2015), S. 97.
[27] Vgl. Fourer et al. (2003), S. xviif.

Synonym für Planung von Aktivitäten mit dem Ziel, unter mehreren Alternativen die bestmögliche Lösung zu finden. Ein klassisches Einsatzgebiet linearer Programmierung ist die Ressourcenzuteilung. Dennoch lassen sich auch zahlreiche weitere praxisrelevante Anwendungsfelder identifizieren, in denen lineare Modelle eingesetzt werden können.[28]

Tatsächlich kann jedes Problem, dessen mathematisches Modell sich in die Struktur eines linearen Programmierungsmodells überführen lässt, als lineares Programmierungsproblem aufgefasst werden. Um solche Probleme selbst bei erheblicher Modellgröße effizient zu lösen, steht mit dem Simplexverfahren eine etablierte und leistungsfähige Lösungsmethode zur Verfügung.[29]

2.3.2 Ganzzahlige und gemischt-ganzzahlige Programmierung

Die ganzzahlige Programmierung stellt eine Erweiterung der linearen Programmierung dar, bei der zusätzlich zu linearen Restriktionen und Zielsetzungen die Bedingung besteht, dass einige oder alle Entscheidungsvariablen ganzzahlige Werte annehmen müssen. Diese scheinbar einfache Modifikation erhöht nicht nur die Ausdrucksstärke solcher Modelle, sondern bringt auch eine erhebliche Zunahme der rechnerischen Komplexität mit sich.

Typische Anwendungen ganzzahliger Modelle finden sich in Bereichen wie Terminplanung, Ressourcenallokation, Supply-Chain-Design, Auktionen und vielen weiteren Gebieten. Die Struktur solcher Modelle basiert, wie bei linearen Programmen, auf Variablen, Nebenbedingungen und einer Zielfunktion. Während die Restriktionen auch hier linear bleiben, liegt die Herausforderung darin, dass ganzzahlige Variablen einen diskreten Lösungsraum erzeugen.

Ein zentrales Problem in der Praxis besteht darin, dass zwei formal ähnliche ganzzahlige Modelle sich im Lösungsvorgang extrem unterschiedlich verhalten können. Während eine Modellierung zur optimalsten Lösung führt, kann eine andere Variante desselben Problems einen längeren Rechenaufwand erfordern. Daraus ergibt sich die Notwendigkeit, nicht nur auf das Modell an sich zu achten, sondern auch auf dessen konkrete Formulierung.

Zu den Erfolgsfaktoren bei der Entwicklung ganzzahliger Modelle zählen unter anderem die kreative Formulierung, die Wahl einer starken Relaxation (also einer linearen Näherungslösung), das Vermeiden von Symmetrien sowie eine gezielte Kontrolle der Anzahl an Variablen und Restriktionen. Zusätzlich können auch die Parameter des Branch-and-

[28] Vgl. Hillier (2014), S. 25.
[29] Vgl. Hillier (2014), S. 26.

Bound-Verfahrens, welches häufig zur Lösung solcher Modelle genutzt wird, angepasst werden, um die Effizienz der Lösungsfindung zu steigern.[30]

Ein wesentliches Hilfsmittel bei der Lösung ganzzahliger Probleme ist die sogenannte lineare Relaxation. Dabei wird das ursprüngliche Integer Programming so angepasst, dass die Ganzzahligkeitsbedingungen entfallen, wodurch ein lineares Programm entsteht. Diese Relaxation ermöglicht die Ableitung von Schranken für die ursprüngliche Zielfunktion. Im Fall eines Minimierungsproblems liefert die Lösung der Relaxation stets einen unteren Schrankenwert, bei Maximierungsproblemen entsprechend einen oberen. Die Relaxation besitzt darüber hinaus weitere nützliche Eigenschaften Ist das lineare Relaxationsproblem unlösbar, so ist auch das ursprüngliche ganzzahlige Problem unlösbar. Sind zudem alle Variablen in der optimalen Lösung des linearen Relaxationsproblems ganzzahlig, so ist diese Lösung zugleich für das ursprüngliche ganzzahlige Problem optimal. Weiterhin gilt, dass sich bei ganzzahligen Zielfunktionskoeffizienten im Minimierungsfall der optimale Wert des ganzzahligen Problems durch Aufrundung des Relaxationswertes abschätzen lässt, während im Maximierungsfall eine Abrundung möglich ist. [31]

Trotz dieser theoretischen Vorteile weist die praktische Umsetzung bestimmte Herausforderungen auf. Besonders bei binären Entscheidungsvariablen, also solchen, die nur die Werte 0 oder 1 annehmen dürfen, kann die Anzahl potenzieller Lösungskombinationen exponentiell wachsen. Ein Modell mit n binären Variablen kann im schlimmsten Fall 2^n Teilprobleme erzeugen, was zu erheblichem Rechenaufwand führen kann. Die Schwierigkeit besteht darin, dass die Schrankenbildung im Branch-and-Bound-Verfahren nicht immer effektiv ist und somit eine große Anzahl an Teilproblemen erzeugt wird. Gerade deshalb ist Kreativität bei der Modellformulierung entscheidend. Integer Programming kann durch gezielte Verwendung ganzzahliger Variablen eine Vielzahl an Strukturen modellieren, die mit rein linearen Ansätzen nicht darstellbar wären. Die Fähigkeit, ganzzahlige Bedingungen gezielt und wirkungsvoll einzusetzen, ermöglicht praxisnahe und leistungsfähige Optimierungsmodelle.[32]

[30] Vgl. Bosch, Trick (2014), S. 67f.
[31] Vgl. Bosch, Trick (2014), S. 70.
[32] Vgl. Bosch, Trick (2014), S. 73.

3. Stand der Forschung

3.1 Aktuelle Ansätze zur KI-gestützten Modellgenerierung

Die Integration von Künstlicher Intelligenz in die mathematische Modellbildung hat in den letzten Jahren erhebliche Fortschritte ermöglicht. Insbesondere leistungsstarke Sprachmodelle wie ChatGPT und Gemini werden zunehmend eingesetzt, um Muster zu erkennen, mathematische Modelle zu optimieren und komplexe Gleichungen zu lösen. Solche Aufgaben waren mit klassischen numerischen Methoden oft nur eingeschränkt möglich. Datengetriebene Methoden des maschinellen Lernens tragen dabei zu einer erhöhten Genauigkeit und Prognosefähigkeit mathematischer Simulationen bei. Moderne KI-Tools unterstützen Forschende heute nicht nur bei der Erklärung mathematischer Zusammenhänge, sondern auch bei der automatisierten Generierung von Programmcode, wodurch sich Modellierungsprozesse deutlich beschleunigen lassen.

Der Einsatz von LLMs wie ChatGPT und Gemini verändert dabei die Herangehensweise an mathematische Modellierung grundlegend. Neben der Fähigkeit, menschliche Sprache zu verstehen und zu generieren, bieten diese Modelle Unterstützung bei der Umwandlung natürlichsprachlicher Beschreibungen in strukturierte, formale Modelle. Gleichzeitig ist in der aktuellen Forschung noch wenig über die systematische Effektivität und Zuverlässigkeit solcher KI-Anwendungen im Vergleich zu traditionellen Modellierungsmethoden bekannt.

Ein zentrales Potenzial dieser Ansätze liegt in der Möglichkeit, komplexe Modellierungsaufgaben zu automatisieren und so auch Anwendern ohne tiefe mathematische Vorkenntnisse Zugang zu modernen Optimierungsmethoden zu eröffnen. Erste Vergleiche zeigen, dass KI-generierte Modelle in vielen Fällen mit etablierten mathematischen Modellen konkurrieren können und insbesondere in der Geschwindigkeit der Modellerstellung Vorteile bieten. Die jüngsten Entwicklungen deuten darauf hin, dass KI-basierte Modellgenerierung in der mathematischen Forschung eine immer größere Rolle spielen wird.[33]

Praktische Untersuchungen unterstreichen diese Entwicklung. Beispielsweise demonstrierten Luzzi et al. (2025) anhand des klassischen Kürzeste-Wege-Problems, dass ChatGPT in der Lage ist, Python-Code für verschiedene Varianten dieses Optimierungsproblems zu generieren und damit konkrete Lösungen für reale Anwendungsfälle bereitzustellen.[34] Im Rahmen des NL4Opt-Wettbewerbs konnte zudem gezeigt werden, dass LLMs wie ChatGPT aus natürlichsprachlichen Optimierungsproblemen formal verwendbare

[33] Vgl. Abosaooda et al. (2025), S. 18ff.
[34] Vgl. Luzzi et al. (2025), S. 1410.

mathematische Modelle erzeugen und damit in der Generierungsaufgabe sogar menschliche Wettbewerbsteilnehmer übertreffen können.[35]

Studien wie die von Liu et al. (2023) belegen zudem, dass ChatGPT nicht nur für die Codegenerierung, sondern auch für Fehlerdiagnose und Übersetzungen zwischen Programmiersprachen eingesetzt werden kann, wobei das Modell in einigen Fällen mit anderen spezialisierten KI-Systemen mithalten oder diese übertreffen konnte.[36]

Darüber hinaus eröffnet der Einsatz generativer KI-Systeme die Möglichkeit, große Datenmengen zu analysieren und daraus innovative Lösungen für Optimierungsaufgaben zu entwickeln. Unternehmen nutzen diese Potenziale zunehmend, um künftige Trends vorherzusagen und operative Prozesse zu optimieren. Gleichzeitig wird die Einbindung von KI-Methoden in klassische Optimierungsverfahren vorangetrieben, um eine höhere Flexibilität, Effizienz und Qualität in der mathematischen Modellierung zu erreichen.[37]

3.2 Forschungsarbeiten im Bereich Optimierung und KI

Ein zentrales Forschungsfeld der letzten Jahre ist der gezielte Einsatz von Künstlicher Intelligenz (KI) und maschinellem Lernen zur Generierung, Optimierung und Weiterentwicklung mathematischer Modelle. Fortschrittliche Frameworks wie TextGrad ermöglichen es, generative KI-Systeme durch LLM-generiertes Feedback systematisch und automatisiert zu verbessern und so den klassischen, heuristisch geprägten Optimierungsprozess zu automatisieren. Im Unterschied zu herkömmlichen Methoden, die auf manuelle Anpassung durch Experten angewiesen sind, nutzt TextGrad die Fähigkeit moderner LLMs, Verbesserungsvorschläge für beliebige Systemkomponenten zu generieren und umzusetzen.[38]

Die Studie von Yuksekgonul et al. zeigt die Leistungsfähigkeit von TextGrad an unterschiedlichen Aufgabenstellungen. In der Codeoptimierung wurde etwa die Erfolgsrate bei der Lösung komplexer LeetCode-Probleme durch TextGrad signifikant gegenüber Zero-Shot-Ansätzen (26 %) und Reflexion (31 %) gesteigert; der Anteil korrekter Lösungen konnte so auf 36 % angehoben werden.[39] Bei wissenschaftlichen Problemlösungen aus GPQA[40] und MMLU[41] Benchmark-Sets verbesserte sich die Performance der zugrunde liegenden LLMs durch iteratives Feedback ebenso. Im Bereich der Prompt-Optimierung konnte TextGrad gezielt Prompts für schwächere, günstigere LLMs so verbessern, dass

[35] Vgl. Ramamonjison et al. (2022), S. 198.
[36] Vgl. Liu, Y. et al. (2023), S. 6.
[37] Vgl. Dubey et al. (2024), S. 8.
[38] Vgl. Yuksekgonul et al. (2025), S. 609 f.
[39] Vgl. Yuksekgonul et al. (2025), S. 612.
[40] Google-proof Question Answering
[41] Massive Multitask Language Understanding

Aufgaben wie logisches Schließen, Zählen und Sortieren effizienter gelöst wurden. Ein besonders praxisnahes Beispiel liefert der medizinische Bereich: Bei der Radiotherapieoptimierung führte das Framework zu klinisch relevanten Verbesserungen hinsichtlich Zielerreichung und Schonung gesunden Gewebes, TextGrad generierte hier iterativ Vorschläge zur Gewichtungsanpassung von Zielgrößen. Auch in komplexen agentenbasierten Systemen und multimodalen Aufgaben ließ sich die Leistung von KI-Systemen durch textbasierte Feedbackschleifen signifikant verbessern, mit Steigerungen bis zu 7,7 %. Ein generischer Ansatz wie TextGrad kann in verschiedenen Kontexten Anwendung finden und leitet damit einen Paradigmenwechsel hin zu domänenübergreifender und erklärbarer Optimierung ein. Die Autoren betonen jedoch auch offene Herausforderungen, etwa den Validierungsbedarf in sensiblen Domänen.[42]

Auch in der technischen Optimierung von Energiesystemen sind KI-Methoden mittlerweile etabliert. So untersuchten Nishad et al. (2024) die Steuerung eines Unified Power Quality Conditioner (UPQC) für Metros, wobei KNN, NARMA-L2 und adaptive PI-Regler miteinander verglichen wurden.[43] Das Ziel war die Minimierung der Total Harmonic Distortion, wobei die KNN-basierte Regelung die stärksten Verbesserungen gegenüber klassischen Ansätzen erzielte, der THD-Wert konnte auf 1,46 % reduziert werden, während das nicht kompensierte System bei 21,74 % lag.[44] Auch die Dynamik der Spannungsregelung war für KNN und NARMA-L2 der klassischen PI-Regelung überlegen, mit Reaktionszeiten von 2 bis 10 ms.[45] Diese Resultate unterstreichen die Überlegenheit KI-basierter Ansätze im dynamischen Betrieb und die Anpassungsfähigkeit gegenüber Veränderungen und Störungen.[46]

Im Bereich des Cloud- und Ressourcenmanagements präsentieren Chaudhary et al. (2025) ein Framework, das KI, Model Order Reduction (MOR) und fortgeschrittene Queueing-Modelle kombiniert, um Ressourcen- und Task-Scheduling in großen verteilten Umgebungen zu optimieren. Das Framework nutzt prädiktive Analysen und intelligente Scheduling-Algorithmen, um auf Lastspitzen zu reagieren und Energieverbrauch zu minimieren.[47] Die Ergebnisse belegen, dass Reaktionszeiten um 50 % sanken, Durchsatz und Ressourcenauslastung signifikant stiegen, während MOR-basierte Modelle den Rechenaufwand um bis zu 80 % verringerten, ohne die Modellgenauigkeit wesentlich zu beeinträchtigen.[48] Der Einsatz von neuronalen Netzen und Reinforcement Learning im

[42] Vgl. Yuksekgonul et al. (2025), S. 613ff.
[43] Vgl. Nishad et al. (2024), S. 1f.
[44] Vgl. Nishad et al. (2024), S. 13f.
[45] Vgl. Nishad et al. (2024), S. 19.
[46] Vgl. Nishad et al. (2024), S. 14.
[47] Vgl. Chaudhary et al. (2025), S. 1f.
[48] Vgl. Chaudhary et al. (2025), S. 17.

Scheduling trägt zu einer weiteren Effizienzsteigerung und Stabilität bei.[49] So entstehen robuste, skalierbare Systemarchitekturen, die das Potenzial moderner KI für Echtzeitoptimierung nutzen.[50]

Auch im Bereich der automatisierten Simulationserstellung gewinnen LLMs an Bedeutung. Die Studie von Obinwanne und Feng (2024) analysiert, wie GPT-4o und Meta-Llama-3-70B für die Generierung von diskreten Ereignissimulationsmodellen in Python eingesetzt werden können.[51] Hier zeigte sich, dass durch gezieltes Prompt Engineering die Erfolgsquote der generierten Codes auf bis zu 90 % gesteigert werden konnte, während unspezifische Prompts deutlich schlechter abschnitten. Trotzdem bleibt menschliche Expertise, besonders bei komplexeren Problemstellungen, für die Qualitätssicherung und Fehlerminimierung unerlässlich. Diese Ergebnisse belegen die aktuelle Leistungsfähigkeit, aber auch die Limitationen von LLMs in der automatisierten mathematischen Modellgenerierung.[52]

Ein besonders anschauliches Beispiel für KI-basierte Prozessoptimierung in der Umwelttechnik liefert die Studie von Ganthavee et al. (2024), die die Entfernung von Methylorange aus Abwasser mit einem 3D-elektrochemischen Verfahren untersucht.[53] KI-Modelle wie KNN, Support Vector Machines und Random Forest wurden zur Modellierung und Optimierung der Prozessparameter genutzt. KNN erzielte die höchste Genauigkeit[54], während Monte-Carlo-Simulationen und Sensitivitätsanalysen die Prognosekraft und Robustheit weiter erhöhten. Die Integration von KI ermöglichte eine effiziente Prozessoptimierung, führte zu höheren Reinigungsleistungen und belegt die Überlegenheit datengetriebener Optimierung im Vergleich zu klassischen Regressionsverfahren.[55]

Ein weiterer Baustein der aktuellen Forschung ist zudem die automatisierte Extraktion von Modellstrukturen aus natürlicher Sprache. Die Studie von Kadıoğlu et al. (2024) stellt mit Ner4Opt einen Ansatz vor, der mittels Named Entity Recognition und LLM-Feintuning die Hauptbestandteile von Optimierungsproblemen automatisch aus Texten erkennt.[56] Die Kombination aus klassischen Feature-Engineering-Methoden und modernen Deep-Learning-Verfahren (wie RoBERTa) lieferte einen F1-Score von 0,919.[57] Auch konnte gezeigt werden, dass Ner4Opt-Annotationen die Qualität KI-generierter Optimierungsmodelle in

[49] Vgl. Chaudhary et al. (2025), S. 8f.
[50] Vgl. Chaudhary et al. (2025), S. 1.
[51] Vgl. Obinwanne, Feng (2025), S. 179f.
[52] Vgl. Obinwanne, Feng (2025), S. 182ff.
[53] Vgl. Ganthavee et al. (2024), S. 2.
[54] Vgl. Ganthavee et al. (2024), S. 17.
[55] Vgl. Ganthavee et al. (2024), S. 26f.
[56] Vgl. Kadıoğlu et al. (2024), S. 262f.
[57] Vgl. Kadıoğlu et al. (2024), S. 276f.

MiniZinc um bis zu 50 % verbessern.[58] Gleichzeitig verdeutlicht die Studie den anhaltenden Forschungsbedarf hinsichtlich Generalisierung und Widerstandsfähigkeit, insbesondere domänenübergreifend.[59]

Schließlich gewinnt die Entwicklung erklärbarer und generalisierbarer KI-Modelle immer mehr an Bedeutung, insbesondere in hochpräzisen industriellen Anwendungen. Die Studie von Luo et al. (2024) stellt ein Grey-Box-Modell vor, das White-Box-Elemente (physikalische Prinzipien) mit Black-Box-Komponenten (datengetriebene Netze) kombiniert.[60] Dieses hybride Modell wurde erfolgreich für die Modellierung eines hochpräzisen Temperaturregelungssystems eingesetzt, wobei die Black-Box-Elemente mittels genetischer Algorithmen an realen Betriebsdaten optimiert wurden.[61] Das Resultat waren höhere Genauigkeit und Stabilität gegenüber klassischen Ansätzen, insbesondere unter veränderten Betriebsbedingungen. Die Studie unterstreicht, dass die Kombination von domänenspezifischem Wissen und datengetriebener Modellanpassung ein zukunftsweisender Ansatz für die Entwicklung präziser, transparenter und adaptiver mathematischer Modelle ist.[62]

3.3 Modellierungsframeworks für mathematische Optimierung

Im Bereich der mathematischen Optimierung haben sich in den letzten Jahren verschiedene leistungsfähige Frameworks etabliert, die eine breite Palette an Modellierungs- und Lösungsmöglichkeiten bieten.

Ein bedeutendes Framework ist MiniZinc, eine Modellierungssprache, die zur Spezifikation von Optimierungs- und Entscheidungsproblemen über Ganzzahlen und reelle Zahlen konzipiert wurde. MiniZinc legt nicht fest, wie ein Problem gelöst wird, sondern übersetzt Modelle mit einem Compiler in verschiedene Formate, die von diversen Solvern wie Constraint Programming, Mixed Integer Linear Programming oder Boolean Satisfiability verarbeitet werden können. Die Sprache erlaubt es, Modelle mit mathematischer Notation wie Quantoren, Summen über Indexmengen und logischen Verknüpfungen zu formulieren und unterstützt darüber hinaus eigene Prädikate und Funktionen zur strukturierten Modellbildung. MiniZinc-Modelle sind meist parametrisierbar und lassen sich flexibel an unterschiedliche Probleminstanzen anpassen. Die Schnittstelle zu verschiedenen Solvern wird

[58] Vgl. Kadıoğlu et al. (2024), S. 285.
[59] Vgl. Kadıoğlu et al. (2024), S. 282–283.
[60] Vgl. Luo et al. (2024), S. 1f.
[61] Vgl. Luo et al. (2024), S. 5ff.
[62] Vgl. Luo et al. (2024), S. 11ff.

über das Zwischenformat FlatZinc realisiert, wobei MiniZinc auch gezielte Steuerung und Feintuning des Lösungsverhaltens durch Annotationen im Modell ermöglicht.[63]

Pyomo ist ein in Python eingebettetes Softwarepaket, das für die mathematische Modellierung und Analyse komplexer Optimierungsanwendungen entwickelt wurde. Es handelt sich um eine algebraische Modellierungssprache, die eng mit dem Python-Ökosystem verknüpft ist und somit die Nutzung zahlreicher Programmiermethoden und Bibliotheken ermöglicht. Pyomo stellt eine einheitliche, hochsprachliche Modellierungssprache bereit, die die Anbindung an verschiedene Solver erleichtert. Im Unterschied zu eigenständigen Modellierungssprachen wie AMPL ist Pyomo direkt in Python integriert und erlaubt es, Modelle als Python-Code zu entwickeln und flexibel mit anderen Datenverarbeitungs- und Analysetools zu kombinieren. Pyomo unterstützt sowohl konkrete als auch abstrakte Modelle, wobei Modellkomponenten wie Zielfunktionen, Nebenbedingungen und Entscheidungsvariablen als Python-Objekte dargestellt werden und so eng mit Datenverarbeitung verbunden werden können.[64]

Zusätzlich im Feld der Optimierung zu nennen ist OR-Tools von Google, eine vielseitige Open-Source-Software, die auf die effiziente Lösung anspruchsvoller kombinatorischer und mathematischer Optimierungsprobleme ausgerichtet ist. OR-Tools bietet eine modulare Architektur und unterstützt die Modellierung in verschiedenen Programmiersprachen. Anwender können aus einer Vielzahl spezialisierter Solver wählen, darunter sowohl kommerzielle als auch Open-Source-Alternativen wie GLOP, CP-SAT, SCIP oder GLPK.[65] Die Lösungsmethoden in OR-Tools reichen von Constraint Programming über lineare und gemischtganzzahlige Optimierung bis hin zu spezialisierten Algorithmen für Routing- und Graphenprobleme. Damit ist OR-Tools nicht nur für klassische lineare Modelle geeignet, sondern auch für anspruchsvolle kombinatorische Aufgabenstellungen und große industrielle Anwendungen.[66]

Ein weiteres etabliertes Framework ist AMPL, das in Kapitel 2.2.2 Modellbildung in AMPL bereits ausführlich behandelt wurde und deshalb an dieser Stelle nicht erneut vorgestellt wird.

3.4 Kritische Würdigung bestehender Ansätze

Die kritische Betrachtung bestehender Ansätze zur mathematischen Optimierung und zur KI-gestützten Modellgenerierung zeigt, dass beide Herangehensweisen jeweils spezifische

[63] Vgl. MiniZinc Team (2022).
[64] Vgl. Bynum et al. (2021), S. 1ff.
[65] Vgl. Google (2025a): About OR-Tools.
[66] Vgl. Google (2025b): Introduction.

Stärken und Schwächen besitzen. Klassische Frameworks wie AMPL und Pyomo werden in der Literatur häufig miteinander verglichen, da sie zu den am weitesten verbreiteten Systemen im Bereich der mathematischen Optimierung zählen. AMPL überzeugt durch seine speziell auf Optimierungsprobleme zugeschnittene, klare und mathematiknahe Syntax, was besonders die Modellformulierung und Pflege vereinfacht. Gerade für Nutzerinnen und Nutzer ohne tiefgehende Programmierkenntnisse gestaltet sich die Arbeit mit AMPL als sehr zugänglich, und auch im Hinblick auf die Performance bei der Generierung großer Modellinstanzen gilt AMPL als Benchmark. Beispielsweise ist AMPL um den Faktor 38 schneller als Pyomo, wenn es um das Laden großer Modelle geht.[67] Darüber hinaus stellt die integrierte Presolving-Funktionalität einen entscheidenden Vorteil dar, da Probleme bereits vor der Solver-Phase effizient vereinfacht werden können, was zu einer deutlichen Beschleunigung des Lösungsprozesses beiträgt und auch dabei hilft, Fehler im Modell frühzeitig zu identifizieren.[68] Pyomo hingegen zeichnet sich durch seine Einbettung in das Python-Ökosystem aus und bietet damit eine hohe Flexibilität und Anschlussfähigkeit an moderne Datenanalyse- und Softwareentwicklungspraktiken. Anwender, die mit objektorientierten Programmiersprachen vertraut sind, können die Möglichkeiten zur Kombination von Modellierung, Datenaufbereitung und Ergebnisvisualisierung innerhalb einer Plattform nutzen. Allerdings zeigt sich in der Praxis, dass Modellierungsaufwand und Codeumfang in Pyomo oft höher ausfallen als in AMPL. Besonders bei sehr großen oder komplexen Modellen stößt Pyomo im Vergleich zu AMPL an seine Leistungsgrenzen, da die Ladezeiten deutlich ansteigen und die Leistungsfähigkeit im Vergleich zu spezialisierten Sprachen geringer ist.[69]

Neben den klassischen Frameworks gewinnen KI-basierte Ansätze zunehmend an Bedeutung. Insbesondere der Einsatz von LLMs wie ChatGPT hat das Potenzial, die mathematische Modellierung und Optimierung grundlegend zu verändern. Die Analyse von Luzzi et al. (2025) zeigt, dass mit ChatGPT generierter Python-Code für Standardprobleme wie das klassische kürzeste-Wege-Problem häufig korrekte und zum Teil sogar effizientere Lösungen als etablierte Referenzalgorithmen liefert, insbesondere bei kleinen und mittelgroßen Instanzen mit überschaubarer Komplexität.[70] Doch sobald die Komplexität des Problems steigt, etwa durch eine höhere Zahl an Knoten oder Farben in Graphen, verschlechtert sich die Effizienz der KI-basierten Lösung. Die Ausführungszeiten steigen deutlich und für sehr komplexe Aufgaben bleiben Lösungen gelegentlich ganz aus. Daher empfehlen die Autoren ausdrücklich, KI-generierte Modelle stets einer kritischen Überprüfung zu

[67] Vgl. Jusevičius et al. (2021), S. 292.
[68] Vgl. Jusevičius et al. (2021), S. 294f.
[69] Vgl. Karbowski, Wyskiel (2021), S.6.; Jusevičius et al. (2022), S. 292, 297.
[70] Vgl. Luzzi et al. (2025), S. 1416.

unterziehen und sie durch Expertenwissen abzusichern, da klassische Algorithmen bei anspruchsvollen Problemen bislang überlegen bleiben.[71]

Auch Abosaooda et al. (2025) bestätigen die Potenziale KI-basierter Modellierungswerkzeuge wie ChatGPT oder Gemini. Sie machen mathematische Modellierung effizienter, flexibler und breiter zugänglich, vor allem für datengetriebene Anwendungen. Durch Automatisierung vieler Modellierungsschritte sinkt der Bedarf an spezialisierten mathematischen Kenntnissen, und die Anpassung an neue Datensituationen wird erleichtert. Gleichzeitig heben die Autoren jedoch hervor, dass der Einsatz solcher KI-Tools immer in enger Abstimmung mit Fachexperten erfolgen muss, um Vertrauen, Validität und Zuverlässigkeit der generierten Modelle sicherzustellen. Obwohl sich der Trend zur Nutzung von KI-Methoden in der mathematischen Modellierung abzeichnet, besteht weiterhin Forschungsbedarf, um die Effektivität und Zuverlässigkeit solcher Systeme in unterschiedlichen Disziplinen zu validieren und weiterzuentwickeln.[72]

In der Praxis, insbesondere im Operations und Supply Chain Management, zeigt sich die Notwendigkeit einer differenzierten Bewertung von generativer KI. Dubey et al. (2024) warnen vor einer unkritischen Begeisterung und betonen, dass die tatsächlichen Potenziale und Grenzen KI-gestützter Anwendungen erst durch theoriegeleitete, empirische Forschung fundiert eingeschätzt werden können. Sie fordern, systematisch zu überprüfen, inwiefern bestehende Methoden und Frameworks den spezifischen Anforderungen des betrieblichen Alltags entsprechen, und plädieren dafür, bestehende Schwächen durch gezielte Weiterentwicklung zu adressieren.[73]

Fosso Wamba et al. (2024) belegen, dass Unternehmen nach der Einführung von Gen-AI und ChatGPT tatsächlich von Effizienzgewinnen und Produktivitätssteigerungen berichten. Jedoch bleibt die Wahrnehmung der Vorteile stark davon abhängig, ob ein Unternehmen bereits Erfahrungen mit diesen Technologien gesammelt hat. Während Nutzer von Verbesserungen berichten, werden Risiken, Unsicherheiten und Barrieren vor allem von Nichtanwendern betont. Die Bedeutung ethischer Aspekte, wie Datenschutz und die Integration mit menschlichen Arbeitsprozesse, bleibt in jedem Fall bestehen, ebenso wie die Notwendigkeit kontinuierlichen organisationalen Lernens, um Risiken zu minimieren und Chancen zu maximieren.[74]

Ein weiterer kritischer Aspekt betrifft die zunehmende Verbreitung von No-Code-AI-Plattformen, die es ermöglichen, KI-basierte Methoden auch ohne tiefgehende

[71] Vgl. Luzzi et al. (2025), S. 1416f.
[72] Vgl. Abosaooda et al. (2025), S. 21.
[73] Vgl. Dubey et al. (2024), S. 15.
[74] Vgl. Fosso Wamba et al. (2023), S. 9.

Programmierkenntnisse zu nutzen. Diese Demokratisierung kann zwar die Entwicklung und Anwendung von KI-Modellen beschleunigen und breiter zugänglich machen, ist aber mit neuen Herausforderungen verbunden. So bleiben grundlegende Probleme wie Datensicherheit, Modell-Interpretierbarkeit und Validierung bestehen und können nicht allein durch Vereinfachung der Entwicklung gelöst werden. Ohne die Einbindung technischer Experten besteht zudem das Risiko, dass Verzerrungen oder Fehler in Modellen nicht rechtzeitig erkannt werden. Die Autoren betonen daher, dass eine enge Zusammenarbeit zwischen Fachabteilungen und Technikspezialisten unverzichtbar ist, um Zuverlässigkeit und Verantwortung beim Einsatz von KI zu gewährleisten. Darüber hinaus können No-Code-AI-Werkzeuge weder die Datenqualität erhöhen noch Verzerrungen im Ausgangsmaterial beseitigen, was fortlaufende Schulungen und Qualitätssicherungsmaßnahmen notwendig macht.[75]

Im betrieblichen Kontext zeigt sich, dass Projekte mit generativer KI häufig noch auf operative Prozessverbesserungen und Kostensenkung ausgerichtet sind. Der Erfolg solcher Systeme hängt maßgeblich von Ressourcen wie Datenkompetenz, technischer Infrastruktur und einer offenen Unternehmenskultur ab. Zu den größten Herausforderungen zählen nach wie vor die Qualität der Daten, technologische Hürden und organisatorische Barrieren. Fosso Wamba et al. (2024) machen darauf aufmerksam, dass die starke mediale Aufmerksamkeit für generative KI oft eine realistische Bewertung der Möglichkeiten erschwert und dass der rasche technologische Wandel die Forschungsergebnisse schnell veralten lassen kann. Weitere Limitationen liegen im explorativen Charakter vieler Studien und der Gefahr, dass von Befragten generative und klassische KI-Konzepte vermischt werden. Sie empfehlen daher, zukünftige Studien erklärender und mit längeren Beobachtungszeiträumen anzulegen, um die tatsächlichen Effekte besser zu verstehen.[76]

Dogru & Keskin, 2020 argumentieren, dass die fortschreitende Automatisierung und der verstärkte Einsatz von KI grundlegende Veränderungen in vielen Branchen auslösen werden, etwa durch effizientere Abläufe, mehr Individualisierung und kürzere Lieferzeiten. Gleichzeitig stehen Unternehmen jedoch vor beträchtlichen Herausforderungen, wie der Sicherstellung von Transparenz und Vertrauenswürdigkeit automatisierter Systeme, dem Aufbau hybrider Arbeitsmodelle und der Entwicklung neuer regulatorischer Rahmenbedingungen zum Schutz sensibler Daten. Besonders kleinere Unternehmen könnten durch die Komplexität und Anforderungen an die IT-Infrastruktur ins Hintertreffen geraten. Hochschulen und Forschungseinrichtungen sind daher gefordert, soziale und technische

[75] Vgl. Sundberg, Holmström (2023), S. 785f.
[76] Vgl. Fosso Wamba et al. (2024), S. 5691.

Kompetenzen stärker miteinander zu verknüpfen, um den neuen Anforderungen des Marktes gerecht zu werden.[77]

Die Gegenüberstellung klassischer Frameworks wie AMPL und Pyomo mit KI-basierten Ansätzen zeigt, dass beide Herangehensweisen spezifische Stärken und Einschränkungen besitzen. Während spezialisierte Modellierungssprachen wie AMPL insbesondere durch Effizienz, Benutzerfreundlichkeit und Performance überzeugen, bietet Pyomo dank seiner Einbettung in das Python-Ökosystem eine hohe Flexibilität, stößt aber bei sehr großen Instanzen an Grenzen.

KI-basierte Systeme wie ChatGPT und Gemini können die mathematische Modellierung und Optimierung vor allem für Standardprobleme deutlich erleichtern und einer breiteren Nutzergruppe zugänglich machen. Allerdings sind sie bei zunehmender Komplexität der Aufgaben bisher klassischen Methoden unterlegen, weshalb der Einsatz stets durch Expertenwissen flankiert werden sollte. Die Einführung von generativer KI und No-Code-Plattformen im betrieblichen Kontext wird vielfach mit Effizienzgewinnen und einer stärkeren Demokratisierung mathematischer Optimierung verbunden. Gleichzeitig bestehen weiterhin Herausforderungen hinsichtlich Datensicherheit, Modellvalidierung, organisatorischer Einbettung und kontinuierlicher Weiterbildung.

Studien machen deutlich, dass der nachhaltige und verantwortungsvolle Einsatz von KI und Optimierungslösungen in der Praxis einen systematischen, interdisziplinären Ansatz erfordert, der technische, organisatorische und ethische Aspekte gleichermaßen berücksichtigt. Für zukünftige Forschung bleibt es zentral, die Effektivität und Zuverlässigkeit neuer Ansätze unter realen Bedingungen differenziert zu evaluieren und bestehende Limitationen gezielt anzugehen.

[77] Vgl. Dogru, Keskin (2020), S. 72f.

4. Methodik

4.1 Forschungsdesign und methodische Einordnung

Die methodische Grundlage dieser Arbeit bildet das Design Science Research (DSR), das in der Informationssystemforschung als etablierter Ansatz zur Entwicklung und Evaluation innovativer IT-Artefakte gilt. DSR ist darauf ausgerichtet, durch die Konstruktion, Anwendung und Reflexion von Artefakten, wie Modellen, Methoden oder Prototypen, sowohl praktischen Nutzen zu schaffen als auch wissenschaftliche Erkenntnisse zu generieren. Das zentrale Prinzip von DSR besteht darin, dass Wissen und Verständnis eines Gestaltungsproblems erst im Prozess der Artefakterstellung und -anwendung entstehen. [78]

Im Sinne der sieben von Hevner et al. (2004) formulierten Leitlinien ,also der Entwicklung eines Artefakts, der Relevanz des adressierten Problems, einer fundierten Evaluation, dem Beitrag zur Forschung, methodischer Strenge, dem Suchprozess nach der besten Lösung sowie der klaren Kommunikation der Ergebnisse, werden in dieser Arbeit ein praxistaugliches Artefakt entwickelt, ein reales, relevantes Problem adressiert, das Vorgehen und die Ergebnisse einer Evaluation unterzogen und die Forschungsergebnisse nachvollziehbar kommuniziert.

Der Forschungsprozess folgt dabei einem iterativen Vorgehen, das sich, in Anlehnung an Hevner et al. (2004), in aufeinander aufbauende Phasen gliedern lässt:

(1) Problem- und Zieldefinition,

(2) Anforderungsanalyse,

(3) Entwurf und Entwicklung eines Prototyps,

(4) empirische Evaluation anhand von Fallstudien sowie

(5) Reflexion und Weiterentwicklung.[79]

Konkret auf diese Arbeit bezogen wurde dieses methodische Rahmenwerk angewendet, um ein KI-basiertes System zu entwickeln, das natürlichsprachlich formulierte Optimierungsprobleme automatisiert in mathematische Modelle überführt. Die iterative Vorgehensweise begann mit einer umfassenden Analyse typischer Optimierungsprobleme, gefolgt von der Ableitung zentraler Anforderungen und der Entwicklung einer modularen Systemarchitektur. Im weiteren Verlauf wurden geeignete Methoden und Technologien ausgewählt und in einem lauffähigen Prototypen implementiert. Die Evaluation erfolgte praxisnah anhand realer Fallstudien, um sowohl die wissenschaftliche Fundierung als auch die

[78] Vgl. Hevner, Chatterjee (2010), S. 5.
[79] Vgl. Hevner et al. (2004), S. 83.

praktische Relevanz des entwickelten Artefakts systematisch zu überprüfen. Die Erkenntnisse aus jeder Phase wurden genutzt, um das System fortlaufend zu verbessern und gezielt weiterzuentwickeln.

4.2 Anforderungsanalyse und Zielsetzung

Ziel dieser Arbeit ist die Entwicklung eines KI-basierten Systems, das in der Lage ist, natürlichsprachliche Optimierungsprobleme automatisiert in formal korrekte mathematische Modelle zu überführen und diese rechnergestützt zu lösen. Damit dieses Ziel erreicht werden kann, müssen verschiedene funktionale und nicht-funktionale Anforderungen erfüllt werden, die sich aus dem aktuellen Stand der Forschung, den spezifischen Eigenschaften mathematischer Optimierungsprobleme und den Anforderungen potenzieller Nutzergruppen ergeben.[80]

Im Vordergrund steht die Fähigkeit des Systems, relevante Informationen aus unstrukturierten, natürlichsprachlichen Problembeschreibungen zu extrahieren und in die Bestandteile formaler Optimierungsmodelle zu überführen.[81] Dazu zählen insbesondere Zielgrößen, Entscheidungsvariablen und Nebenbedingungen, die je nach Formulierung unterschiedlich explizit oder implizit enthalten sein können.[82] Eine weitere zentrale Anforderung besteht darin, die identifizierten Elemente in eine etablierte mathematische Modellierungssprache zu transformieren, sodass das resultierende Modell vollständig und korrekt abgebildet und von entsprechenden Optimierungsverfahren verarbeitet werden kann.[83] Für eine hohe Praxistauglichkeit sind zudem Robustheit im Umgang mit unvollständigen oder fehlerhaften Eingaben, Modularität und Erweiterbarkeit des Gesamtsystems sowie eine benutzerfreundliche und transparente Gestaltung wesentlich.[84]

Nicht zuletzt sind Effizienz der Modellgenerierung, Datenschutz und Nachvollziehbarkeit aller Systemschritte zu gewährleisten, um das System auch im betrieblichen Kontext und mit sensiblen Daten einsetzen zu können.[85]

Die genannten Anforderungen bestimmen die Zielsetzung dieser Arbeit und bilden die konzeptionelle Grundlage für die im Folgenden entwickelte Systemarchitektur sowie für die Auswahl und Integration geeigneter Methoden. Die konkrete technische Umsetzung dieser Anforderungen wird in Kapitel 5 ausführlich dargestellt.

[80] Vgl. Luzzi et al. (2025), S. 1416f.
[81] Vgl. Devlin et al. (2019), S. 4171.
[82] Vgl. Fourer et al. (2003), S. 6.
[83] Vgl. Fourer et al. (2003), S. xvii; Bynum et al. (2021), S. 3.
[84] Vgl. Doshi-Velez, Kim (2017), S. 2f.
[85] Vgl. Abosaooda et al. (2025), S. 21; Luzzi et al. (2025), S. 1417.

4.3 Auswahl und Begründung der eingesetzten Methoden

Die Auswahl der in dieser Arbeit verwendeten Methoden orientiert sich sowohl am aktuellen Stand der Forschung als auch an den in Abschnitt 4.2 beschriebenen Anforderungen an ein KI-basiertes System zur automatisierten Modellgenerierung. Ziel ist es, einen Ansatz zu realisieren, der sowohl leistungsfähig als auch flexibel und praxistauglich ist.

Moderne LLMs wie die GPT-Reihe sind darauf ausgelegt, komplexe Aufgabenstellungen in natürlicher Sprache zu analysieren und durch sogenannte Prompts spezifische Aufgaben, etwa die Strukturierung, Extraktion oder Umformulierung von Information, auszuführen. Studien zeigen, dass diese Modelle, insbesondere im Kontext von Prompt-based Learning, in der Lage sind, unstrukturierte Texte zuverlässig in strukturierte Formate zu überführen.[86] In der aktuellen Forschung wird hervorgehoben, dass die Wirksamkeit von LLMs in domänenspezifischen Anwendungen maßgeblich von der zielgerichteten Gestaltung der Prompts abhängt. Ein gezieltes Prompt Engineering ermöglicht nicht nur eine bessere Kontrolle der Ergebnisse, sondern unterstützt auch die Übertragbarkeit der Modelle auf spezifische Problemstellungen.[87]

Zur mathematischen Modellierung wurde die Sprache AMPL gewählt, da sie eine deklarative und flexible Modellierung von Optimierungsproblemen erlaubt und in der mathematischen Optimierung als Standardwerkzeug breite Anwendung findet.[88] Dadurch ist eine präzise und solverunabhängige Umsetzung der generierten Modelle möglich. Die zugrundeliegende Systemarchitektur wurde modular konzipiert, um die spätere Erweiterung um alternative Komponenten oder Technologien zu erleichtern und eine nachhaltige Wartbarkeit sicherzustellen. Zusätzlich wurde Wert auf eine benutzerfreundliche Gestaltung, Transparenz der Verarbeitungsschritte und Nachvollziehbarkeit der Ergebnisse gelegt, um das System auch für den praktischen Einsatz, etwa im Lehr- oder Unternehmenskontext, nutzbar zu machen.

Die konkrete technische Ausgestaltung und Umsetzung der gewählten Methoden wird im folgenden Kapitel 5 im Detail dargestellt.

[86] Vgl. Min et al. (2023), S. 10.
[87] Vgl. Fatawi et al. (2024), S. 442.
[88] Vgl. Fourer et al. (2003), S. xviif.

4.4 Abgrenzungen und Limitationen

Im Rahmen dieser Arbeit wurde der Fokus gezielt auf die automatisierte Generierung formaler Optimierungsmodelle aus natürlichsprachlichen Problembeschreibungen gelegt. Dabei ergeben sich Abgrenzungen und Limitationen, die sich aus den gewählten methodischen Ansätzen und technischen Komponenten ableiten.

Das entwickelte System ist auf klassische Optimierungsprobleme mit deterministischer Struktur ausgerichtet. Es unterstützt insbesondere lineare, ganzzahlige und gemischt-ganzzahlige Optimierungsmodelle, wie sie in Lehre und Praxis typischerweise vorkommen. Nicht abgedeckt werden nichtlineare, stochastische oder dynamische Modelle sowie Problemstellungen mit mehreren Entscheidungsebenen. Die Fokussierung auf die Modellierungssprache AMPL begrenzt die Übertragbarkeit des Ansatzes auf andere Modellierungsumgebungen, auch wenn eine Anpassung grundsätzlich möglich wäre.

Das zugrundeliegende LLM (GPT-4o mini) arbeitet auf Basis allgemeiner Trainingsdaten und ist nicht speziell auf den Bereich mathematische Optimierung oder branchenspezifische Fachsprache trainiert. In der Fachliteratur ist bekannt, dass LLMs insbesondere bei komplexen, mehrdeutigen oder unvollständigen Texteingaben an ihre Grenzen stoßen können. Die Zuverlässigkeit der automatisierten Modellübersetzung hängt daher maßgeblich von der Präzision und Klarheit der Nutzereingaben ab.[89]

Das System ist modular aufgebaut, jedoch derzeit auf die Anbindung an AMPL und einen bestimmten Optimierungssolver beschränkt. Weitere Solver oder alternative Modellierungstools können prinzipiell integriert werden, wurden aber nicht umgesetzt. Die Nutzung externer Sprachmodelle erfordert eine stabile API-Anbindung und setzt die Verfügbarkeit der entsprechenden Dienste voraus. Datenschutzaspekte und der Umgang mit sensiblen Daten wurden nicht im Detail untersucht und sind bei einem möglichen Praxiseinsatz zu berücksichtigen.

Die aktuelle Systemversion bietet keine dialogorientierte Interaktion oder automatische Validierung der generierten Modelle. Unklare, mehrdeutige oder fehlerhafte Eingaben werden nicht automatisch erkannt oder rückgemeldet. Die Überprüfung der fachlichen und mathematischen Korrektheit der Ergebnisse liegt daher in der Verantwortung der Nutzer.

Die Systembewertung basiert auf ausgewählten Fallstudien aus Lehre und Praxis. Eine umfassende Validierung für großskalige industrielle Anwendungen oder sehr komplexe Optimierungsaufgaben war nicht Gegenstand dieser Arbeit.

[89] Vgl. Liu et al. (2023), S. 7; Min et al. (2023), S. 9.

5. Entwicklung des Ansatzes

5.1 Architektur des entwickelten Systems

Die Architektur des im Rahmen dieser Masterarbeit entwickelten Systems zur KI-basierten Generierung mathematischer Optimierungsmodelle ist das Ergebnis eines interdisziplinären Entwicklungsprozesses, bei dem moderne Methoden der Künstlichen Intelligenz mit klassischen Verfahren der mathematischen Modellierung und Optimierung kombiniert wurden. Der zugrundeliegende Anspruch ist es, ein generisches und zugleich leistungsfähiges Werkzeug zu schaffen, das in der Lage ist, eine Vielzahl praktischer Optimierungsaufgaben weitgehend automatisiert und ohne spezielle Modellierungsvorkenntnisse des Anwenders zu bearbeiten.

Im Mittelpunkt der Architektur steht eine modulare, pipelineartige Struktur, welche die unterschiedlichen Aufgabenbereiche klar voneinander trennt, aber über definierte Schnittstellen miteinander verbindet. Der Gesamtprozess beginnt mit der Eingabe einer Optimierungsaufgabe durch den Nutzer. Hierfür steht eine flexible, textbasierte Nutzerschnittstelle zur Verfügung, die es ermöglicht, Problemstellungen in natürlicher Sprache einzugeben. Diese Schnittstelle ist darauf ausgelegt, so benutzerfreundlich wie möglich zu sein und die Hemmschwelle für die Nutzung mathematischer Optimierungstools erheblich zu senken. Der Nutzer beschreibt beispielsweise ein Produktions-, Transport- oder Zuordnungsproblem so, wie er es auch einem Kollegen oder Berater schildern würde. Dies ist eine der zentralen Innovationen des Systems, da die klassische Trennung zwischen Problemformulierung und Modellierung aufgehoben wird.

Nach der Eingabe der Problemstellung erfolgt eine automatische Klassifikation des Problems. Ein eigens entwickeltes Regelwerk analysiert dazu den Text und sucht nach Schlüsselwörtern, numerischen Angaben und charakteristischen Strukturen, um die zugrundeliegende mathematische Problemklasse zu identifizieren. Die Unterscheidung zwischen beispielsweise Transport-, Set-Covering-, Produktionsplanungs- oder Verschnittproblemen ist entscheidend, da sie die Auswahl und die Struktur des zu generierenden Modells maßgeblich beeinflusst. Abhängig vom erkannten Problemtyp wird ein Prompt für das zugrundeliegende LLM erzeugt, welches in der Lage ist, mathematische Optimierungsmodelle zu formulieren.

Die Einbindung des LLM (GPT-4o von OpenAI) stellt das Herzstück des Systems dar. Hierbei wird die natürliche Spracheingabe in einen für das LLM optimierten Prompt überführt, der mit spezifischen Anweisungen angereichert ist. Diese Anweisungen legen fest, dass das resultierende AMPL-Modell ausschließlich die mathematische Struktur, also Mengen-, Parameter-, Variablen- und Nebenbedingungsdefinitionen, enthalten soll. Zugleich

wird explizit untersagt, dass Instanzdaten direkt im AMPL-Modell hinterlegt werden; diese werden stattdessen im Python-Teil mittels der Schnittstelle amplpy gesetzt. Dadurch ist sichergestellt, dass das System universell für verschiedene Problemstellungen und Dateninstanzen einsetzbar bleibt. Die Architektur folgt damit dem Prinzip der strikten Trennung von Modellstruktur und Daten, was eine Wiederverwendbarkeit und Flexibilität garantiert.

Nach Generierung des Prompts wird dieser an das LLM gesendet, welches daraufhin einen ausführbaren Code generiert. Die Rückgabe des Modells erfolgt als formatierter Python-Codeblock, der sowohl das AMPL-Modell als Multiline-String als auch den Python-Steuerungsteil enthält. Mittels regulärer Ausdrücke wird aus der Antwort des LLM exakt der relevante Codeteil extrahiert. Dies ist ein kritischer Schritt, da die Konsistenz und Ausführbarkeit des resultierenden Codes direkt von der Qualität der LLM-Antwort abhängt. Die Architektur sieht daher eine zusätzliche Fehlerbehandlungs- und Preprocessing-Stufe vor. In dieser werden potenziell fehleranfällige AMPL-Konstrukte, wie etwa unzulässige Mengen- oder Parameterzuweisungen im Modellteil oder die Verwendung nicht unterstützter Funktionen, erkannt und korrigiert oder gegebenenfalls dem Nutzer zur manuellen Überprüfung signalisiert. Durch diese Plausibilitätsprüfung werden Stabilität und Fehlerresistenz des Systems signifikant erhöht.

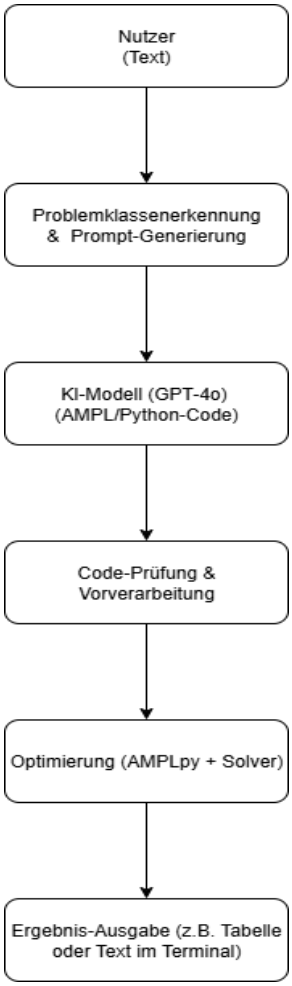


Abbildung 1: Schematische Übersicht der Systemarchitektur und -pipeline des entwickelten KI-basierten Modellgenerators.

Im Anschluss an die Generierung und Überprüfung des Modellcodes wird dieser in einer temporären Datei gespeichert und automatisch ausgeführt. Hierbei kommt die Python-Bibliothek amplpy zum Einsatz, welche eine Schnittstelle zur leistungsstarken Optimierungssoftware AMPL und zu verschiedenen Solver-Engines wie CBC oder HiGHS bildet. Alle erforderlichen Instanzdaten, etwa Maschinenkapazitäten, Marktgrenzen, Lagerkosten oder Zugehörigkeiten bei Set-Covering-Problemen, werden dabei im Python-Teil in Form von Dictionaries oder Listen definiert und per Methodenaufruf (ampl.set[...], ampl.param[...]) an das AMPL-Modell übergeben. Das ermöglicht nicht nur eine saubere Trennung von Modell und Daten, sondern auch eine einfache Wiederverwendung und Parametrisierung für unterschiedliche Szenarien.

Die Optimierung selbst erfolgt vollautomatisch im Hintergrund. Das System wählt anhand der Problemstruktur und -größe einen geeigneten Solver und übergibt das aufbereitete Modell an die

Optimierungsengine. Nach erfolgreicher Lösung werden die Ergebnisse über die amplpy-Schnittstelle ausgelesen und in einer nach Anwenderbedürfnissen gestalteten Form ausgegeben. Die Ergebnisaufbereitung ist dabei so gestaltet, dass sie sowohl für einfache Textausgaben (z. B. Produktionspläne, Transportmengen, Zuordnungen) als auch für weitergehende Visualisierungen (z. B. Tabellen, Diagramme, Gantt-Charts) erweitert werden kann. Typischerweise werden alle relevanten Entscheidungsvariablen sowie die Zielgröße, wie beispielsweise Gesamtkosten, Gesamtgewinn oder die minimale Anzahl benötigter Ressourcen, in klarer, tabellarischer Form präsentiert.

Ein besonderes Merkmal der Systemarchitektur ist ihre Erweiterbarkeit. Neue Problemklassen können durch das Ergänzen entsprechender Regeln im Klassifikationsmodul sowie spezifischer Prompts für das LLM einfach integriert werden. Ebenso ist die Anbindung weiterer Solver oder Visualisierungskomponenten ohne grundlegende Änderungen an der Gesamtstruktur möglich.

Die Gesamtsystemarchitektur ist in Abbildung 1 visualisiert. Sie zeigt die einzelnen Prozessschritte, von der Nutzereingabe über die automatische Klassifikation und KI-gestützte Modellgenerierung bis hin zur Ergebnisaufbereitung.

5.2 Datenquellen und -vorverarbeitung

Die Datenverarbeitung und die Integration der Datenquellen im entwickelten System basieren vollständig auf einer klaren Trennung zwischen Modellstruktur und Daten, wie sie in der Architektur und insbesondere im aktuellen Stand des Programmcodes umgesetzt ist. Die Implementierung folgt einem generischen Ansatz, bei dem das mathematische Optimierungsmodell unabhängig von konkreten Daten formuliert wird und sämtliche Instanzdaten erst im Python-Teil programmatisch ergänzt werden. Das folgende Kapitel beschreibt, wie Datenquellen eingebunden, verarbeitet und für den Optimierungsprozess aufbereitet werden.

Im Rahmen des entwickelten Systems stellt die Nutzereingabe in Form eines frei formulierten Freitextes die zentrale Datenquelle dar. Bereits in der Architektur und im aktuellen Programmdesign ist vorgesehen, dass die zu lösende Optimierungsaufgabe in der Variable `user_problem` als natürlichsprachlicher Textblock erfasst wird. Die Problemstellung kann dabei flexibel und ohne Vorgabe einer bestimmten Syntax oder Datenstruktur formuliert werden, beispielsweise als Produktionsplanungs-, Transport- oder Zuordnungsproblem. Dies ermöglicht eine große Bandbreite an Anwendungsszenarien und erleichtert insbesondere auch Anwendern ohne spezifische Vorkenntnisse im Bereich der mathematischen Modellierung den Zugang zum System.

Die Eingabe erfolgt direkt im Python-Skript oder über ein geeignetes Texteingabefeld. Durch die Nutzung eines offenen Freitextfeldes stehen unterschiedliche Optionen zur Formulierung zur Verfügung, sodass verschiedene Nutzergruppen adressiert werden können.

Darüber hinaus ist das System darauf ausgelegt, perspektivisch auch andere Schnittstellen zur Problemaufnahme zu integrieren, etwa Web-Frontends, grafische Benutzeroberflächen oder Sprachschnittstellen (Speech-to-Text). Die Aufnahme des Nutzerproblems als Freitext stellt somit nicht nur eine anwenderfreundliche, sondern auch eine zukunftsfähige Lösung dar. Im weiteren Verlauf dient der eingegebene Freitext als primärer Input für die Modellgenerierung, indem daraus sämtliche für die mathematische Formulierung und Lösung notwendigen Informationen extrahiert werden.

5.3 Struktur und Komponenten des KI-basierten Ansatzes

Die in Abschnitt 5.2 beschriebene Erfassung der Optimierungsaufgabe als Freitext bildet den Ausgangspunkt der Systempipeline. Im Folgenden werden die Systemkomponenten und deren Zusammenspiel im Rahmen der automatisierten Modellgenerierung dargestellt.

Das in dieser Arbeit vorgestellte System zur automatisierten Generierung mathematischer Optimierungsmodelle basiert auf einer konsequent modularen Architektur. Die einzelnen Komponenten sind so gestaltet, dass sie unterschiedliche Ebenen der Problemverarbeitung abdecken und zugleich in enger Interaktion einen konsistenten, vollständig automatisierten Modellierungsprozess ermöglichen.

Im Zentrum des Ansatzes steht das Ziel, den gesamten Prozess von der Erfassung des Optimierungsproblems in natürlicher Sprache bis zur Präsentation der Lösungsergebnisse so zu gestalten, dass keine manuellen Zwischenschritte mehr notwendig sind. Damit wird nicht nur die Zugänglichkeit mathematischer Optimierung für Nutzer ohne spezielles Fachwissen erhöht, sondern auch die Fehleranfälligkeit und der Aufwand traditioneller, manueller Modellierungsprozesse reduziert.

Der Einstiegspunkt in die Systempipeline ist die Erfassung des Nutzerproblems als Freitext. Diese freie Formulierung wird zunächst von einer Analysekomponente interpretiert, die relevante Strukturen, Zielsetzungen und Randbedingungen extrahiert und klassifiziert. Diese Klassifikation erfolgt dabei nicht über eine starre Zuordnung zu vordefinierten Modellklassen, sondern durch eine adaptive Erkennung relevanter Muster, Modellierungskonzepte und Problemtypen. Das System erkennt somit flexibel, ob es sich etwa um ein Zuordnungs-, Transport-, Zeitplanungs- oder Zuschnittproblem handelt.

Im nächsten Schritt werden die so gewonnenen Informationen in eine strukturierte, für KI-basierte Modellgenerierung geeignete Repräsentation überführt. Diese umfasst insbesondere die explizite Trennung zwischen der abstrakten Modellstruktur und den spezifischen Instanzdaten, wobei die Modellstruktur allgemeingültig gehalten und von konkreten Datenbezügen freigehalten wird. Ziel ist es, Modelle zu erzeugen, die unabhängig von einzelnen Dateninstanzen wiederverwendbar und durch Austausch der Instanzdaten universell einsetzbar sind. Dieses Prinzip ist zugleich Grundlage für die künftige Anbindung externer Datenquellen und die Automatisierung wiederkehrender Optimierungsaufgaben.

Zentrales Bindeglied der Architektur ist ein generisches Modul zur Aufgabenstrukturierung und -übersetzung. Es übernimmt die Aufgabe, die zuvor extrahierten und strukturierten Problemmerkmale so aufzubereiten, dass sie für die KI-gestützte Modellgenerierung weiterverarbeitet werden können. Hierzu zählt insbesondere die Abbildung mathematischer Grundelemente (z. B. Mengen, Parameter, Entscheidungsvariablen, Nebenbedingungen) sowie die eindeutige Definition von Modellierungskonventionen, beispielsweise hinsichtlich erlaubter Modellierungstechniken oder der geforderten Ergebnisdarstellung.

Ein Element der Systemarchitektur bildet das KI-gestützte Modul zur automatisierten Modellgenerierung. Diese Komponente fungiert als Vermittlungsinstanz zwischen der natürlichsprachlichen Beschreibung des Nutzerproblems und der formalen, mathematisch exakten Modellstruktur. Ziel ist es, auf Grundlage der strukturierten, zuvor extrahierten Problemmerkmale ein exekutierbares Optimierungsmodell zu erzeugen, das sämtliche vorgegebenen Modellierungskonventionen berücksichtigt. Dazu zählen insbesondere Vorgaben hinsichtlich der Modellstruktur, der zulässigen Variablentypen, der mathematischen Nebenbedingungen sowie der erforderlichen Mechanismen zur Ergebnispräsentation. Die Generierung erfolgt stets regelkonform und anpassungsfähig, sodass die resultierenden Modelle unabhängig von der jeweiligen Aufgabenstellung sowohl formal korrekt als auch direkt ausführbar sind. Die automatisierte Auswertung und Darstellung der Optimierungsergebnisse erfolgt ebenfalls systemgesteuert und kann flexibel an unterschiedliche Problemtypen sowie individuelle Ausgabewünsche angepasst werden.

Die Systemarchitektur folgt dabei strikt dem Prinzip der Modularität. Jede Komponente, von der Problemklassifikation über die strukturierte Aufgabenrepräsentation bis zur Modellgenerierung und Ergebnisaufbereitung, ist über wohldefinierte Schnittstellen mit den jeweils folgenden Modulen verbunden. Dies ermöglicht sowohl eine gezielte Erweiterung um neue Funktionalitäten und Modellierungsmethoden als auch eine problemlose Integration zusätzlicher Daten- und Ausgabemöglichkeiten.

Das hier beschriebene konzeptionelle Zusammenspiel der Systemmodule schafft somit die Grundlage für die im nächsten Abschnitt dargestellte technische Umsetzung der automatisierten Modellgenerierung.

5.4 Implementierung der Modellgenerierung

Die Implementierung des entwickelten Systems ist vollständig KI-basiert und modular gestaltet, wobei jeder Schritt des automatisierten Modellgenerierungsprozesses systematisch durch spezialisierte Komponenten abgedeckt wird. Nach der Aufnahme der Optimierungsaufgabe als Freitext, wird der eingegebene Problembeschreibungstext (`user_problem`) zunächst von einer regelbasierten Analysekomponente ausgewertet. Die Extraktion und Klassifikation erfolgt dabei nicht direkt durch vorgegebene Kategorien, sondern durch die Einbettung des Nutzertexts in ein umfassendes Prompt-Template. Dieses Template fungiert als flexibles Regelwerk, das dem zugrundeliegenden LLM die relevante Interpretation der Aufgabenstellung ermöglicht.

Konkret wird im Prompt-Template festgelegt, welche mathematischen Strukturen, Nebenbedingungen und Variablentypen für die jeweilige Problemstellung zu berücksichtigen sind. Eindeutige Anweisungen sorgen beispielsweise dafür, dass eine strikte Trennung zwischen Modellstruktur und Instanzdaten eingehalten wird. Datenzuweisungen im AMPL-Modell sind dabei untersagt und werden ausschließlich im Python-Teil des Codes vorgenommen. Das Regelwerk steuert zudem die Verwendung spezifischer Modellierungsbegriffe und -konzepte, wie Mengen, Parameter und ganzzahlige Entscheidungsvariablen, und schreibt die Einhaltung zentraler Modellierungskonventionen wie den Verzicht auf spezielle AMPL-Funktionen (`ord()`, `prev()`, `first()` etc) vor. Darüber hinaus enthält das Prompt-Template klare Instruktionen zur Formatierung und Ausgabe der Entscheidungsvariablen, um eine konsistente und nachvollziehbare Ergebnispräsentation zu gewährleisten.

Die technische Umsetzung dieser Prompt-Erstellung erfolgt im Code mittels eines f-string-Templates, in das der Nutzertext und die ergänzenden Anweisungen dynamisch eingebettet werden. Der Prompt-Generator erstellt so eine Aufgabenbeschreibung, die nicht nur den Freitext des Nutzers, sondern auch alle erforderlichen technischen Vorgaben für das LLM beinhaltet. Die Funktion dieser Komponente ist es, die Präzision der Modellbildung sowie die Konsistenz und Automatisierbarkeit der nachfolgenden Prozessschritte sicherzustellen, von der Code-Generierung über die Datenintegration bis zur Ergebnispräsentation.

Im Zentrum des KI-basierten Systems steht das LLM, in der vorliegenden Implementierung GPT-4o von OpenAI, das über die OpenAI-API eingebunden wird. Nach Übergabe des spezifisch aufgebauten Prompts produziert das LLM einen vollständigen, syntaktisch

ausführbaren Python-Code, der das AMPL-Modell als Multiline-String sowie sämtliche erforderlichen Datenzuweisungen und Steuerungsanweisungen enthält. Die Initialisierung der OpenAI-API sowie die Konstruktion des API-Clients sind integraler Bestandteil der technischen Umsetzung. Die Kommunikation zwischen System und LLM erfolgt asynchron, sodass auch größere oder komplexere Prompts effizient verarbeitet werden können.

Nach Eingang der LLM-Antwort wird der relevante Python-Code gezielt extrahiert. Hierbei kommen reguläre Ausdrücke zum Einsatz, um ausschließlich den direkt ausführbaren Code für die Weiterverarbeitung bereitzustellen. Ein wichtiger Schritt ist die anschließende Validierung: Die Funktion `is_ampl_code_problematic` prüft den generierten Code regelbasiert auf potenziell problematische AMPL-Konstrukte, darunter verbotene Datenzuweisungen (`data;, :=`), nicht zugelassene Funktionen, sowie direkte Modell-Daten-Integrationen, die der Systemarchitektur widersprechen würden. Mithilfe expliziter Prüfmechanismen wird sichergestellt, dass ausschließlich Code zur weiteren Ausführung gelangt, der den definierten Systemanforderungen entspricht. Fehlerhafte oder unvollständige Modellteile werden entweder automatisiert korrigiert oder an den Nutzer zur Überprüfung zurückgemeldet.

Die Integration und Instanziierung der konkreten Problemdaten erfolgt in der aktuellen Systemarchitektur ausschließlich im Python-Teil des Codes. Alle für die jeweilige Optimierungsaufgabe relevanten Instanzdaten, wie Mengen, Parameter, Kapazitäten, Nachfragewerte oder Kosten, werden als Python-Datenstrukturen, als Listen oder Dictionaries, gepflegt und mittels Methoden wie `ampl.set[...]` und `ampl.param[...]` ins AMPL-Modell eingebracht. Das generierte mathematische Modell bleibt dabei stets generisch und kann für unterschiedlichste Instanzen wiederverwendet werden, während jeweils nur die Instanzdaten angepasst werden müssen.

Darüber hinaus ermöglicht dieser Ansatz perspektivisch die nahtlose Integration externer Datenquellen, wie etwa Datenbanken, CSV- oder Excel-Dateien sowie Web-APIs, in den Optimierungsprozess. Auch die Validierung der Datenintegration kann vor dem Start der Optimierung automatisiert erfolgen, was die Transparenz und Nachvollziehbarkeit der Lösung weiter verbessert.

Die Auswahl und Steuerung der Solver-Engine wird im System nicht unmittelbar im Framework-Code, sondern als Bestandteil der Prompt-Generierung direkt in den generierten Python-Code integriert. Das Prompt-Template spezifiziert dazu, dass im Python-Code vor dem Aufruf von `ampl.solve()` die gewünschte Solver-Engine, beispielsweise mittels `ampl.setOption('solver', 'cbc')` oder alternativ `ampl.setOption('solver', 'highs')`, festgelegt wird. Durch diese Architektur ist es möglich, verschiedene Open-Source-Solver flexibel zu nutzen, ohne dass Anpassungen am Kernsystem notwendig werden. Neue oder

alternative Solver können durch geringfügige Änderungen am Prompt-Template in den generierten Code eingebracht werden, was die Skalierbarkeit und Erweiterbarkeit des Systems weiter erhöht.

Nach vollständiger Generierung und erfolgreicher Prüfung des Modellcodes wird dieser, einschließlich der Solver-Konfiguration, in einer temporären Datei gespeichert und mittels eines systemseitigen Aufrufs (`subprocess.run(["python", filename])`) automatisiert ausgeführt. Die Optimierung und Lösung des mathematischen Modells geschieht somit im generierten Python-Code, der im Hintergrund gestartet wird.

Die Ergebnisaufbereitung ist flexibel gehalten und an keine festen Vorgaben gebunden. In der aktuellen Systemausgestaltung erfolgt die Ausgabe sämtlicher Entscheidungsvariablen und Zielwerte nach der erfolgreichen Optimierung dynamisch im Python-Teil des Codes. Die konkrete Form der präsentierten Ergebnisse variiert je nach Modelltyp und Aufgabenstellung. Während bei Transportproblemen beispielsweise die gelieferten Mengen und die minimalen Gesamtkosten ausgegeben werden, stehen bei Produktionsplanungsaufgaben produzierten Stückzahlen und Gesamtgewinn im Fokus. In anderen Fällen werden etwa die optimale Anzahl an Zuschnitten, die Verteilung von Pflegepersonal auf Abteilungen und Schichten oder Verschnittmengen und deren Verteilung tabellarisch präsentiert.

Die konkrete Form der Ausgabe ergibt sich direkt aus der jeweiligen Struktur des mathematischen Modells und ist nicht fest im System kodiert. Neue Problemklassen lassen sich daher ohne grundlegende Anpassungen an der Ausgabelogik integrieren. Darüber hinaus ist das System offen für Erweiterungen. Die Ergebnisse können bei Bedarf nicht nur als Text oder Tabelle, sondern auch als Diagramm, Gantt-Chart oder in anderen Formaten bereitgestellt werden. Damit bleibt die Präsentation der Resultate stets anpassbar und erweiterbar, je nach Nutzeranforderung und Aufgabenstellung.

Typische Beispielausgaben sind etwa die tabellarische Darstellung der Produktionsmengen und Gewinne, Transportpläne mit zugehörigen Liefermengen und Gesamtkosten oder der optimale Personaleinsatz im Krankenhaus. Das Zusammenspiel der einzelnen technischen Komponenten folgt einer klar definierten Pipeline, in der die Daten und Ergebnisse schrittweise verarbeitet und für die jeweils nächste Systemkomponente aufbereitet werden.

Die Ausgabe umfasst sowohl den Wert der Zielfunktion (Gesamtkosten, Gesamtgewinn) als auch die detaillierten Werte der relevanten Entscheidungsvariablen. Im Beispiel einer Operationsplanung kann dies folgendermaßen aussehen:

```
Generierter Code gespeichert unter: C:\Users\...\tmpufndmojq.py
Starte Ausführung...
---

CBC 2.10.3 optimal, objective -21500
0 nodes, 0 iterations, 0.014 seconds
Objective Value (Total Profit): 21500.0
Number of Herz operations: 0.0
Number of Orthopädie operations: 0.0
Number of Viszeral operations: 0.0
Number of Unfallchirurgie operations: 5.0
Number of HNO operations: 5.0
```

Weitere Beispiele für typische Ausgaben aus unterschiedlichen Optimierungsproblemen sind:

- Produktionsplanung:

```
Optimal solution:
Produce P1: 3.0 units
Produce P2: 6.0 units
Total Profit: 210.0
```

- Transportproblem:

```
Optimaler Transportplan:
Fabrik A zu Lager W1: 0.0 Einheiten
Fabrik A zu Lager W2: 5.0 Einheiten
...
Minimale Gesamtkosten: 100.0 Euro
```

- Zweidimensionaler Zuschnitt:

```
Optimale Anzahl an Rohplatten: 23.0
Anzahl der Zuschnitte für Produkt A: 20.0
...
```

- Personaleinsatzplanung im Krankenhaus:

```
Objective value (Total Nurses): 69.0
Nurses in department I, shift 1: 6.0
...
```

Diese Beispiele zeigen, dass das System die Ergebnispräsentation an die jeweilige Problemstruktur anpasst und somit eine benutzerorientierte Darstellung ermöglicht.

5.5 Integration von Optimierungsverfahren

Die Integration der Optimierungsverfahren bildet eine Schnittstelle zwischen der im System generierten mathematischen Modellstruktur und der algorithmischen Lösungsfindung. Der Ansatz des entwickelten Systems folgt hierbei dem Prinzip der Modularität und Austauschbarkeit. Die eigentliche Modellierung, die im AMPL-Format erfolgt, bleibt stets von der Auswahl und Konfiguration des Optimierungsalgorithmus entkoppelt.

Die Umsetzung der Solver-Anbindung erfolgt im Rahmen des Python-Teils mittels der Schnittstelle `amplpy`. Diese Bibliothek ermöglicht eine nahtlose Übergabe der generierten Modelle sowie sämtlicher Instanzdaten an leistungsfähige Open-Source-Solver wie CBC und HiGHS. Die Auswahl des gewünschten Solvers wird dabei explizit im Code über die Methode `ampl.setOption('solver', 'cbc')`bzw. `# alternativ: ampl.setOption ('solver', 'highs')` realisiert. Durch diese explizite Steuerung können für unterschiedliche Aufgabenstellungen jeweils geeignete Verfahren genutzt werden, etwa branch-and-cut für ganzzahlige Probleme oder spezialisierte Simplex- und Barrier-Algorithmen für lineare und gemischt-ganzzahlige Modelle.

Nach Abschluss der Modellgenerierung und Datenintegration werden sämtliche Modell- und Instanzinformationen an die AMPL-Engine übertragen. Die Initialisierung und Steuerung des Optimierungslaufs erfolgt anschließend vollautomatisch über das gewählte Optimierungsverfahren. Die Trennung von Modellstruktur, Instanzdaten und Lösungsalgorithmus bewirkt, dass die Lösungsschritte unabhängig voneinander weiterentwickelt oder angepasst werden können.

Im konkreten Ablauf bedeutet dies, dass das System nach dem Setzen aller Instanzdaten das Optimierungsmodell speichert, den Solver initialisiert und den Berechnungsvorgang startet. Während der Lösungsberechnung werden keine Benutzereingriffe benötigt, sodass ein vollständig automatisierter und reproduzierbarer Workflow gewährleistet ist.

Die Wahl einer offenen und modularen Solver-Integration stellt sicher, dass das System jederzeit um neue Optimierungsverfahren, spezifische Solver-Features oder alternative algorithmische Strategien erweitert werden kann. Die explizite Definition der Solver-Optionen im Python-Teil des Codes erlaubt es, je nach Problemstruktur und Anforderungen unterschiedliche Verfahren zu nutzen, Parameter zu variieren oder individuelle Präferenzen umzusetzen, ohne dass Änderungen an der grundlegenden Modellstruktur erforderlich sind.

Zudem ist es durch die Nutzung der `amplpy`-Schnittstelle möglich, auch künftige technische Entwicklungen, wie Cloud-basierte Solver, Parallelisierungsoptionen oder domänenspezifische Spezialalgorithmen, in den Lösungsprozess zu integrieren.

Nach erfolgreichem Abschluss der Optimierung werden die Resultate, darunter die Werte der Zielfunktion sowie aller relevanten Entscheidungsvariablen, direkt über die `amplpy`-API ausgelesen. Diese Werte bilden die Grundlage für die anschließende Ergebnisaufbereitung, wie sie im vorangegangenen Kapitel beschrieben wurde. Die strukturierte Übergabe der Lösungen aus dem Optimierungslauf ermöglicht eine vielseitige Auswertung, Präsentation und Weiterverarbeitung, sei es in tabellarischer, grafischer oder textueller Form.

6. Anwendungsbeispiele und Evaluation

6.1 Auswahl und Beschreibung der Fallstudien

Für die Evaluation des entwickelten KI-basierten Systems wurden unterschiedliche praxisnahe Fallstudien ausgewählt und als Optimierungsaufgaben implementiert. Die Auswahl der Fallstudien erfolgte mit dem Ziel, eine möglichst große Bandbreite klassischer Optimierungsprobleme aus der industriellen und betrieblichen Praxis abzudecken. Die Mehrheit dieser Aufgabenstellungen basiert auf bewährten Lehr- und Übungsbeispielen aus dem Lehrmaterial von Prof. Dr. T. Mellouli an der Martin-Luther-Universität Halle-Wittenberg. Ergänzt wird dieses Spektrum durch gezielt entwickelte eigenständige Fallbeispiele, um moderne betriebliche Kontexte und neue Anwendungsfelder abzubilden.

1. Personaleinsatzplanung im Krankenhaus[90]

Innerhalb dieser Fallstudie steht die optimale Zuteilung von Pflegepersonal auf drei Abteilungen (Intensivstation, Orthopädie, Entbindungsstation) und drei Schichten innerhalb eines Krankenhauses im Fokus. Für jede Kombination aus Abteilung und Schicht sind sowohl Mindest- als auch Maximalwerte für den Personaleinsatz vorgegeben. Darüber hinaus gelten übergeordnete Mindestanforderungen für die gesamte Anzahl der Pflegekräfte pro Abteilung sowie pro Schicht.

Modellstruktur:

- Variablen: Anzahl der Pfleger pro Abteilung und Schicht
- Parameter: Mindest-/Maximalwerte siehe unten; Mindestgesamtbedarf: I: ≥13, II: ≥32, III: ≥22; Schicht 1: ≥26, 2: ≥24, 3: ≥19
- Nebenbedingungen: Einhaltung aller Unter- und Obergrenzen je Kombination und insgesamt
- Ziel: Minimierung der Gesamtzahl eingesetzter Pflegekräfte bei vollständiger Abdeckung aller Anforderungen

Dieses kombinatorische Zuteilungsproblem ist prototypisch für die Personaleinsatzplanung im Gesundheitswesen und verlangt eine ganzzahlige Modellierung.

[90] Vgl. Mellouli (2015).

Tabelle 1: Mindest- und Maximalbesetzung der Personaleinsatzplanung im Krankenhaus.

Schicht/Abt	I	II	III
1	6-8	11-12	7-12
2	4-6	11-12	7-12
3	2-4	10-12	5-7

Quelle: Eigene Darstellung.

2. OP-Planung in einer Klinik

Quelle: eigene Fallstudie in Anlehnung an Mellouli

In diesem Szenario plant eine Klinik die Durchführung verschiedener Operationen (Herz, Orthopädie, Viszeral, Unfallchirurgie, HNO) innerhalb einer Woche mit dem Ziel, den Gewinn zu maximieren. Jede Operation unterscheidet sich in Zeitbedarf, benötigtem Personal (Pflegekräfte, Ärzte), Gewinn und maximaler Patientenzahl. Für die Planung stehen nur begrenzte Ressourcen wie OP-Säle, OP-Stunden, Pflegekräfte und Ärzte zur Verfügung.

Modellstruktur:

- Variablen: Anzahl der durchzuführenden Operationen je Art
- Parameter: Siehe Tabelle unten; Ressourcen: 4 OP-Säle (12 OP-Tage, 60 Std), 15 Pflegekräfte, 10 Ärzte
- Nebenbedingungen: Geplante Operationen je Art ≤ max. Patienten; Gesamte OP-Zeit ≤ 60 Std;
- Personalrestriktionen beachten
- Ziel: Maximierung des Gesamtgewinns der Klinik

Das Modell bildet ein klassisches Ressourcenallokationsproblem mit mehreren, sich überlappenden Restriktionen ab.

Tabelle 2: Übersicht OP Planung.

OP-Art	Gewinn (€)	OP-Zeit (h)	Pflegebedarf	Ärztebedarf	max. Patienten
Herz	5000	6	4	3	4
Orthopädie	3500	4	3	2	8
Viszeral	4200	5	3	2	6
Unfallchirurgie	2500	3	2	1	10
HNO	1800	2	1	1	8

Quelle: Eigene Darstellung.

3. Verschnittoptimierung bei Papierrollen[91]

Diese Fallstudie befasst sich mit dem sogenannten Zuschnittproblem. Eine Papierfabrik muss Kundenaufträge für Rollen unterschiedlicher Breiten und Längen mit Standardrollen (zwei verfügbare Breiten) beliefern. Die Herausforderung liegt darin, die zugeschnittenen Längen aus jeder Rollenbreite so zu bestimmen, dass alle Aufträge abgedeckt sind und der Materialverschnitt minimiert wird.

Modellstruktur:

- Variablen: Zugeschnittene Länge (m) aus Rollenbreite für jeden Auftrag
- Parameter: Auftragsdaten: siehe Tabelle 3; Verfügbare Standardrollen: Typ 1: 20 m, Typ 2: 10 m
- Nebenbedingungen: Für jeden Auftrag: Summe [zugeschnittene Länge × Rollenbreite] ≥ (Auftragsbreite × Auftragslänge); Kleben in Längsrichtung erlaubt
- Ziel: Minimierung der Gesamtverschnittmenge (Papierfläche, die nicht verwendet werden kann)

Das Problem verlangt die effiziente Zuweisung von Standardmaterial auf kundenindividuelle Spezifikationen und ist typisch für viele Fertigungsindustrien.

Tabelle 3: Auftragsdaten der Papierrollen.

Auftrag	Breite (m)	Länge (m)
A	5	15.000
B	7	35.000
C	9	20.000

Quelle: Eigene Darstellung.

4. Mehrperiodige Produktionsplanung mit Wartung[92]

Quelle: eigene Vereinfachung nach Mellouli

Die Ausgangslage dieser Fallstudie besteht darin, dass die Fertigung von drei Produkten auf zwei Maschinen über drei Monate geplant wird. Jedes Produkt hat einen spezifischen Deckungsbeitrag und unterschiedliche Bearbeitungszeiten je Maschine. Die Maschinenkapazitäten sind pro Monat beschränkt, im zweiten Monat reduziert eine Wartung die verfügbare Zeit einer Maschine zusätzlich. Zu berücksichtigen sind Nachfrageschwankungen je Produkt und Monat, Lagerkapazitäten (mit Kosten), sowie geforderte Mindestendbestände.

[91] Vgl. Mellouli (2021).
[92] Vgl. Mellouli (2021).

Modellstruktur:

- Variablen: Produktions-, Lager- und Verkaufsmenge je Produkt/Monat
- Parameter: Produkte: P1, P2, P3; Deckungsbeiträge (€): 10, 8, 7; Bearbeitungszeit pro Stück/Maschine: M1: (1,2,1), M2: (2,1,2), Maschinenkapazität/Monat: M1: 60, M2: 50 (Monat 2: 30); Lagerkapazität: 10 Stück/Produkt/Monat, Lagerkosten: 1 €/Stück/Monat; Anfangslager: 0, Endlager (Monat 3): ≥3 Stück/Produkt
- Nebenbedingungen: Maschinenzeiten/Monat inkl. Wartung beachten; Verkaufsmenge ≤ Nachfrage; Lagerhaltungskapazität beachten; Endlager ≥ 3 Stück/Produkt
- Ziel: Maximierung des Gesamtgewinns unter Berücksichtigung der Lagerkosten

Das Modell vereint Aspekte der Produktions-, Lager- und Ressourcenplanung und illustriert typische Planungsprobleme in der Industrie.

Nachfrage pro Monat:

Tabelle 4: Nachfrage pro Monat.

	Monat 1	**Monat 2**	**Monat 3**
P1	10	12	8
P2	15	10	12
P3	7	8	9

Quelle: Eigene Darstellung.

5. Netzwerkoptimierung: Flughafen-Logistik

Hier wird die Versorgung mehrerer Flughafen-Terminals mit Waren von drei Lagerstandorten als Netzwerkflussproblem modelliert. Jedes Lager und jedes Terminal hat Kapazitätsgrenzen; nicht jede Verbindung ist erlaubt. Für jede Relation ist eine Transportkapazität festgelegt. Ziel ist es, alle Terminals vollständig zu versorgen und dabei den Gesamttransport zu minimieren.

Modellstruktur:

- Variablen: Transportmengen von jedem Lager zu jedem Terminal
- Parameter: Lagerkapazität: Nord: 100, Süd: 80, Ost: 70 Paletten/Tag; Terminalbedarf: T1: 60, T2: 50, T3: 70, T4: 40 Paletten/Tag
- Nebenbedingungen: Liefermenge je Lager ≤ Lagerkapazität; Gesamtmenge je Terminal ≥ Terminalbedarf; Transport je Route ≤ Routen-Kapazität
- Ziel: Minimierung der Gesamttransportmenge (Summe aller gelieferten Paletten)

Kapazitätsmatrix:

Die konkreten Modellparameter lauten:

Tabelle 5: Maximale Lieferkapazität der Lagerstandorte.

Lager	Kapazität (Einheiten)
Nord	100
Süd	80
Ost	70

Quelle: Eigene Darstellung.

Tabelle 6: Festgelegte Nachfrage der Terminals.

Terminal	Nachfrage (Einheiten)
T1	60
T2	50
T3	70
T4	40

Quelle: Eigene Darstellung.

Tabelle 7: Transportkapazitäten der erlaubten Routen.

	T1	T2	T3	T4
Nord	60	0	40	0
Süd	30	50	0	30
Ost	0	30	40	30

Quelle: Eigene Darstellung.

Dieses Fallbeispiel ist ein klassisches Minimum-Cost-Flow-Problem mit zusätzlichen Kapazitäts- und Netzwerkrestriktionen, wie sie in der Logistik und im Supply Chain Management regelmäßig auftreten.

6.2 Ergebnisse der automatisierten Modellgenerierung

Nach der detaillierten Analyse und Beschreibung der ausgewählten, praxisrelevanten Fallstudien im vorherigen Kapitel werden in diesem Abschnitt die Ergebnisse der automatisierten Modellgenerierung dargestellt. Im Zentrum stehen die Überprüfung der Anwendbarkeit und Effizienz der entwickelten KI-basierten Modellierungsmethode im Hinblick auf unterschiedliche Problemstellungen.

Zur exemplarischen Bewertung der Leistung der automatisierten Modellgenerierung und Lösungsfindung wurde das Netzwerk-Transportproblem am Flughafen in mehrfachen

Durchläufen analysiert. Hierbei wurden sowohl die Zeiten für die KI-basierte Modellerstellung als auch die Solverlaufzeiten erhoben und statistisch ausgewertet (siehe Tabelle 13).

Auch sämtliche weiteren betrachteten Fallstudien konnten zuverlässig und jeweils in sehr kurzer Zeit gelöst werden. Da in keinem Fall kritische Laufzeitprobleme auftraten, wurde auf eine detaillierte tabellarische Auswertung sämtlicher Laufzeiten zugunsten der Übersichtlichkeit verzichtet und die Darstellung exemplarisch auf das Netzwerkmodell beschränkt.

Im Folgenden werden zunächst die Lösungsergebnisse der jeweils betrachteten Modelle im Detail präsentiert. Anschließend erfolgt die Diskussion der ermittelten Leistungskennzahlen der automatisierten Modellgenerierung.

1. Personaleinsatzplanung im Krankenhaus

Das Ziel der automatisierten Optimierung bestand darin, die minimale Gesamtanzahl an Pflegekräften so auf Abteilungen und Schichten zu verteilen, dass sämtliche betrieblichen Mindest- und Maximalbesetzungen erfüllt werden. Dafür wurde auf Basis einer natürlichsprachigen Aufgabenbeschreibung automatisch ein ganzzahliges Optimierungsmodell generiert, im Python-Teil mit den relevanten Falldaten belegt und anschließend mit dem CBC-Solver gelöst.

Die durchschnittliche Laufzeit für die automatische Modellerstellung betrug etwa 13 Sekunden (Spannweite: 9,8–19,5 Sekunden). Die mittlere Rechenzeit für das eigentliche Lösen des Optimierungsproblems lag bei unter 0,05 Sekunden. Somit konnte der gesamte Workflow, von der Problemformulierung bis zur Lösung, in deutlich unter 20 Sekunden durchlaufen werden.

Die optimale Lösung umfasst eine minimale Gesamtanzahl von 69 Pflegekräften, verteilt wie folgt auf die einzelnen Abteilungen und Schichten:

Tabelle 8: Optimallösung der Verteilung der Pflegekräfte.

Schicht	Abt. I	Abt. II	Abt. III	Summe Schicht
1	6	11	9	26
2	5	11	8	24
3	2	12	5	19
Summe Abt.	13	34	22	69

Quelle: Eigene Darstellung.

2. OP-Planung in einer Klinik

Das Ziel dieses Beispiels ist es, unter Berücksichtigung aller verfügbaren Ressourcen, darunter OP-Zeit, Pflegepersonal, Ärzte sowie Obergrenzen für jede OP-Art, den wöchentlichen Gesamtgewinn der Klinik zu maximieren.

Das entwickelte System berechnete die folgende optimale Operationsverteilung:

Tabelle 9: Optimale Operationsverteilung.

OP-Art	Anzahl Operationen
Herz	0
Orthopädie	0
Viszeral	0
Unfallchirurgie	5
HNO	5

Quelle: Eigene Darstellung.

Der maximal erzielbare Gewinn beträgt 21.500 € pro Woche.

Die durchschnittliche Laufzeit für die automatische Modellerstellung lag bei etwa 13 Sekunden (Spannweite: 9–23 Sekunden), die mittlere Solverlaufzeit stets unter 0,07 Sekunden.

Bemerkenswert ist, dass ausschließlich die weniger ressourcenintensiven Operationen (Unfallchirurgie und HNO) geplant werden, während komplexere Eingriffe aufgrund der begrenzten Kapazitäten im Optimum nicht realisiert werden. Dies verdeutlicht, wie das System Ressourcenrestriktionen konsistent abbildet und ökonomisch sinnvolle, an die Kapazitätsgrenzen angepasste OP-Pläne erstellt.

Zu beachten ist, dass der Solver-Output das Ergebnis mit negativem Vorzeichen (−21.500) ausgab, was auf einen typischen Vorzeichenfehler in der automatisch generierten Zielfunktion zurückzuführen ist. Für die Auswertung wurde das Vorzeichen entsprechend angepasst. Dieser Aspekt unterstreicht die Notwendigkeit einer kritischen Überprüfung automatisch erzeugter Modellformulierungen.

3. Verschnittoptimierung bei Papierrollen

Zur Minimierung des Verschnitts in der Papierproduktion wurde das in Abschnitt 6.1 beschriebene Optimierungsmodell implementiert und mit den realen Auftrags- und Rollenbreiten parametriert. Die Zielsetzung bestand darin, für jede Auftragsposition die optimale Aufteilung der Zuschnittlängen aus den verfügbaren Standardrollen zu bestimmen, sodass die geforderte Liefermenge pro Auftrag erfüllt und der anfallende Verschnitt minimiert wird.

Das Modell basiert auf der Formulierung, dass jedem Auftrag eine oder mehrere Zuschnittlängen aus den verfügbaren Rollenbreiten zugeordnet werden können. Entscheidend ist dabei die Einhaltung der Flächenbedingung. Für jeden Auftrag muss mindestens die bestellte Fläche (Breite × Länge) geliefert werden. Die Variablen des Modells geben an, wie viele Laufmeter aus welcher Rollenbreite für welchen Auftrag zugeschnitten werden.

Die Optimierung wurde mit Hilfe eines linearen Programms gelöst. Die Ergebnisse zeigen, dass in der vorliegenden Aufgabenstellung eine vollständige Abdeckung aller Aufträge ausschließlich durch den Einsatz der 10 m breiten Standardrolle (Rollen-Typ 2) optimal ist. Der Grund hierfür liegt im geringeren relativen Verschnitt je Quadratmeter im Vergleich zur 20 m breiten Rolle. Für keinen Auftrag ist der Zuschnitt aus der breiteren Rolle von Vorteil, sodass Typ 1 nicht genutzt wird.

Im Detail ergeben sich die folgenden optimalen Zuschnittlängen:

- Auftrag A: 7.500 m aus Typ 2
- Auftrag B: 24.500 m aus Typ 2
- Auftrag C: 18.000 m aus Typ 2

Der gesamte Verschnitt summiert sich somit auf 129.000 m². Das Modell stellt sicher, dass jeder Auftrag exakt die benötigte Fläche erhält und gleichzeitig der Gesamtverschnitt über alle Aufträge hinweg minimal ist.

4. Mehrperiodige Produktionsplanung mit Wartung

Im Rahmen der Fallstudie zur mehrperiodigen Produktionsplanung wurde das KI-basierte System auf ein praxisnahes Problem mit drei Produkten, zwei Maschinen und drei Planungsperioden angewendet. Die Aufgabenstellung umfasste Maschinenkapazitäten, Wartungsphasen, Nachfrageschwankungen, Lagerkosten sowie einen geforderten Mindestendbestand.

Das System generierte automatisch ein lauffähiges Optimierungsmodell, in dem sämtliche Restriktionen und Zielsetzungen korrekt umgesetzt wurden. Die Analyse der Modellergebnisse zeigt, dass das System in der Lage ist, Produktion und Lagerhaltung so zu steuern, dass sowohl alle Nachfrage- und Endbestandsbedingungen erfüllt als auch die Gewinnmaximierung unter Einhaltung der Kapazitätsgrenzen erreicht wird. Besonders hervorzuheben ist, dass in Monaten mit geringerer Nachfrage gezielt vorproduziert und gelagert wird, um Engpässe in Wartungsphasen abzufedern und die Lagerkosten möglichst gering zu halten.

Zwei typische Ausprägungen der Lösung traten auf:

(1) Zulässige Optimallösung:

Sofern alle Kapazitätsrestriktionen, insbesondere im Wartungsmonat, exakt eingehalten werden, liefert das System eine vollständige, restriktionskonforme Produktions- und Lagerplanung mit maximalem Gesamtgewinn:

Tabelle 10: Optimallösung Produktions- & Lagerplanung.

Monat	Produkt	Produktion	Verkauf	Lager am Monatsende
1	P1	12	10	2
2	P1	10	12	0
3	P1	11	8	3
1	P2	16	15	1
2	P2	10	10	1
3	P2	14	12	3
1	P3	5	5	0
2	P3	0	0	0
3	P3	7	4	3

Quelle: Eigene Darstellung.

Der maximal erzielbare Gesamtgewinn beträgt in diesem Fall 646 €. Das Modell demonstriert, wie durch gezielte Vorproduktion und Lagerhaltung Kapazitätsengpässe überwunden und die ökonomisch optimale Lösung erzielt werden kann.

(2) Unzulässige Lösung bei Überlast:

Wird jedoch beispielsweise durch zu hohe Nachfrage oder zu hohe Endlageranforderungen die Maschinenkapazität in einer Periode überschritten, so gibt das System eine Warnung aus und generiert eine Notlösung:

```
Warning:

    presolve: constraint MachineCapacity['M2',2] cannot hold: body <= 30 cannot
    be >= 40; difference = -10

...

Total Profit: 149.0
```

Tabelle 11: Unzulässige Lösung bei Überlast.

Monat	Produkt	Produktion	Lagerbestand am Monatsende
1	P1	10	0
2	P1	0	0
3	P1	0	0
1	P2	0	0
2	P2	0	0
3	P2	0	0
1	P3	7	0
2	P3	0	0
3	P3	0	0

Quelle: Eigene Darstellung.

In diesem Fall werden nur Anfangsbestände produziert und verkauft. Eine valide Mehrperiodenplanung ist mathematisch nicht möglich, da die Kapazitätsrestriktionen verletzt werden.

Die durchschnittliche Laufzeit für die automatische Modellerstellung lag bei rund 14 Sekunden, die Solverlaufzeit jeweils unter 0,04 Sekunden.

In Gesamtbetrachtung verdeutlicht das Beispiel, dass das KI-basierte Optimierungssystem auch komplexe mehrperiodige Produktionsprobleme zuverlässig automatisiert lösen kann, sofern die Eingabedaten die Einhaltung aller Kapazitätsrestriktionen erlauben. Gleichzeitig wird deutlich, dass bereits geringfügige Änderungen an Restriktionsparametern, insbesondere bei knappen Engpassressourcen, dazu führen können, dass das Modell nicht mehr lösbar ist. Es empfiehlt sich daher, die Machbarkeit der Eingabedaten vor der Optimierung automatisiert zu prüfen und im Zweifel Anpassungen vorzunehmen.

5. Netzwerkoptimierung: Flughafen-Logistik

In dieser Fallstudie wurde die optimale Verteilung von Waren aus drei Lagerstandorten („Nord“, „Süd“ und „Ost“) zu vier Terminals (T1–T4) eines Flughafens untersucht. Jeder Lagerstandort verfügt über eine maximale Lieferkapazität, jedes Terminal hat eine festgelegte Nachfrage. Die Transportkapazitäten auf den einzelnen Relationen sind durch infrastrukturelle Gegebenheiten beschränkt; Transporte auf nicht explizit erlaubten Routen sind nicht möglich.

Das Ziel des Modells ist, die Gesamtzahl der transportierten Einheiten zu minimieren, wobei alle Nachfragen erfüllt und alle Kapazitätsgrenzen eingehalten werden müssen.

Das mathematische Optimierungsmodell liefert eine zulässige Verteilung, die die Gesamtnachfrage deckt. Aufgrund mehrfach optimaler Lösungen können sich die Einzelwerte zwischen den Solverläufen geringfügig unterscheiden, so kann beispielsweise der Wert für „Ost→T4“ je nach Lauf zwischen 10 und 30 variieren, sofern die übrigen Restriktionen eingehalten werden.

Tabelle 12: Optimallösung Terminal-Warenlieferung.

	T1	**T2**	**T3**	**T4**	**Summe Lager**
Nord	60	0	40	0	100
Süd	0	50	0	30	80
Ost	0	0	30	10	40
Summe Terminal	60	50	70	40	220

Quelle: Eigene Darstellung.

Laufzeiten:

Die Analyse von 20 Durchläufen zeigt, dass die mittlere Modellgenerierungszeit bei 14,9 Sekunden liegt (Median: 14,1 s, Minimum: 10,8 s, Maximum: 24,4 s). Die Optimierung im Solver erfolgt äußerst schnell und zuverlässig, mit einer durchschnittlichen Solverlaufzeit von 0,041 Sekunden (Median: 0,043 s, Minimum: 0,025 s, Maximum: 0,078 s). Daraus wird ersichtlich, dass die Hauptlaufzeit durch die KI-basierte Modellerstellung bestimmt wird, während das eigentliche mathematische Optimierungsmodell für diese Problemgröße innerhalb von Sekundenbruchteilen, und somit nahezu in Echtzeit, gelöst werden kann.

Tabelle 13: Laufzeiten des Flughafen-Logistiknetzwerks.

Durchlauf	Modellgenerierungs-zeit (s)	Solverlaufzeit (s)
1	21.38	0.0776
2	16.44	0.0265
3	13.82	0.0490
4	11.42	0.0404
5	12.39	0.0254
6	14.12	0.0497
7	14.27	0.0279
8	10.84	0.0550
9	14.49	0.0347
10	13.48	0.0316
11	13.83	0.0444
12	16.68	0.0430
13	14.05	0.0527
14	14.05	0.0329
15	17.95	0.0427
16	17.22	0.0448
17	24.40	0.0248
18	15.54	0.0477
19	13.71	0.0359
20	13.09	0.0469

Quelle: Eigene Darstellung.

6.3 Diskussion der Ergebnisse

Die automatisierte Generierung und Lösung von Optimierungsmodellen anhand der ausgewählten Fallstudien ermöglichte eine umfassende Evaluation der Leistungsfähigkeit des entwickelten KI-basierten Systems. Die Ergebnisse belegen, dass das System in der Lage ist, eine Vielzahl unterschiedlicher Optimierungsprobleme aus der Praxis korrekt zu erkennen, zu modellieren und mit Hilfe etablierter Solver effizient zu lösen. Im Folgenden werden die Stärken, Schwächen sowie die Limitationen des Ansatzes kritisch analysiert und mit bestehenden Forschungsansätzen verglichen.

Stärken und positive Beobachtungen

Ein wesentliches Ergebnis der Evaluation ist, dass die automatisierte Modellgenerierung für klar strukturierte Aufgabenstellungen, wie sie in der Personaleinsatzplanung, der mehrperiodigen Produktionsplanung mit Wartung oder klassischen Netzwerkflussproblemen vorkommen, recht zuverlässig funktioniert. Das entwickelte KI-System ist in der Lage, typische LP-Probleme zu erfassen, die Modellstruktur korrekt aufzubauen und sinnvolle Nebenbedingungen zu implementieren.

Hervorzuheben ist die Geschwindigkeit der Modellbildung. Für die meisten der betrachteten Anwendungsfälle lag die Zeitspanne für die automatische Generierung eines mathematischen Modells im niedrigen Sekundenbereich. Auch die eigentliche Lösung der Modelle durch einen Optimierungssolver erfolgte, selbst bei komplexeren Problemstellungen, in weniger als einer Sekunde. Dadurch ergibt sich im Vergleich zur konventionellen, manuellen Modellierung und Kodierung eine deutliche Zeitersparnis. Der Einsatz des Systems eignet sich somit besonders für Anwendungsfälle, bei denen regelmäßig ähnliche Problemtypen gelöst werden müssen und der Modellierungsaufwand minimal gehalten werden soll.

Ein weiterer Vorteil des Ansatzes liegt in der Benutzerfreundlichkeit. Die Modellgenerierung kann von Anwendern auch ohne tiefergehende Kenntnisse in mathematischer Optimierung oder spezieller Modellierungssprachen wie AMPL genutzt werden. Die einzige Voraussetzung ist die Bereitstellung einer strukturierten, möglichst präzisen Aufgabenbeschreibung. Für typische Fälle genügt es, diese Beschreibung gezielt an die jeweilige Fragestellung anzupassen. Das System ist somit auch für Laien oder gelegentliche Nutzer im Bereich der mathematischen Optimierung einsetzbar.

Die Implementierung spezifischer Restriktionen wie Maschinenwartungen, Kapazitäts- oder Nachfragegrenzen sowie Mindest- und Maximalanforderungen wurde durch das System in der Regel korrekt erfasst.

Schwächen und Herausforderungen

Die Evaluierung hat jedoch auch deutliche Schwächen und Grenzen des entwickelten Ansatzes offengelegt. Zunächst ist der Systemerfolg in starkem Maße von der Klarheit, Präzision und Vollständigkeit der Aufgabenbeschreibung abhängig. Viele Aufgabenstellungen aus klassischen Fallstudien oder Lehrbüchern mussten zunächst sprachlich oder strukturell umgearbeitet werden, damit sie von der KI ohne Fehler interpretiert werden konnten. Dies betrifft insbesondere mehrdeutige Formulierungen, unklare Restriktionen oder komplex verschachtelte Bedingungen.

Eine der größten Herausforderungen bestand in der Umsetzung der AMPL-Syntax. Die KI generierte häufig Modelle, die inhaltlich korrekt waren, aber kleinere oder größere Syntaxfehler aufwiesen, etwa bei der Indizierung von Variablen, beim Umgang mit Datenparametern, bei der Formulierung von Summen oder bei der Behandlung von Zeitperioden und mehrdimensionalen Indizes. Solche Fehler führten entweder zu Modellen, die vom Solver nicht akzeptiert wurden, oder zu inhaltlich falschen Modellformulierungen. In mehreren Fällen mussten die automatisch erzeugten Modelle manuell nachbearbeitet werden, um eine lauffähige und inhaltlich korrekte AMPL-Implementierung zu erhalten.

Hinzu kommt, dass die KI, wie viele aktuelle LLMs, bei identischen user problems mitunter unterschiedliche Modellergebnisse erzeugte. Dies wurde insbesondere dann beobachtet, wenn das Modell mehrfach ausgeführt oder die Aufgabenstellung leicht variiert wurde. Die Ergebnisse waren nicht immer konsistent. Manchmal fehlten einzelne Randbedingungen, in anderen Fällen wurden Modellelemente übersehen oder fehlerhaft implementiert. Dies beeinträchtigt die Zuverlässigkeit und macht eine Qualitätssicherung erforderlich.

Typische Fehler und Limitationen

Eine zentrale Limitation des Systems besteht in der Unfähigkeit, komplexe, mehrstufige oder ungewöhnliche Restriktionen immer korrekt und vollständig zu erfassen. Beispielsweise kam es bei Produktionsplanungsaufgaben mit vielen Zeitperioden, Indizes oder produktabhängigen Restriktionen häufig zu Fehlern in der Modellstruktur. Besonders schwierig sind Aufgaben, bei denen Bedingungen auf mehreren Ebenen verknüpft sind, zum Beispiel Ressourcenrestriktionen in Abhängigkeit von mehreren Parametern, oder Aufgaben, die von klassischen LP-Standards abweichen.

Bei der automatisierten Modellgenerierung traten typische Fehler auf, darunter eine falsche oder fehlende Indizierung von Variablen, etwa durch das Nichtberücksichtigen aller Zeitperioden, sowie das Übersehen oder die Fehlinterpretation spezifischer Nebenbedingungen wie Kapazitäts- oder Lagerrestriktionen. Auch eine fehlerhafte Parametrierung, insbesondere bei der Übertragung tabellarischer Daten in Modellparameter, konnte beobachtet werden. Darüber hinaus führten Syntaxfehler in der Formulierung der AMPL-Modelle zu Fehlermeldungen im Solver. In einigen Fällen zeigte sich zudem eine nicht einheitliche Modellierung bei mehrfacher Ausführung derselben Aufgabenstellung.

In Einzelfällen konnte das KI-System die Modellstruktur nicht vollständig oder korrekt abbilden, sodass eine manuelle Korrektur zwingend erforderlich war. Dies betraf vor allem Fälle mit unvollständigen oder unpräzise formulierten Aufgabenstellungen, aber auch komplexe Produktions- oder Netzwerkprobleme.

Abhängigkeit von der Aufgabenbeschreibung und Usability

Ein Ergebnis der Evaluation ist, dass die Qualität der automatisch generierten Modelle sehr stark von der Formulierung und Detailliertheit der Aufgabenbeschreibung abhängt. Je klarer, strukturierter und vollständiger das user problem formuliert ist, desto zuverlässiger und vollständiger erfolgt die Modellgenerierung. Unklare oder unvollständige Eingaben führen dagegen zu Missverständnissen, zu fehlenden Restriktionen oder zu fehlerhaften mathematischen Modellen. Daher empfiehlt es sich in der Anwendung, die Problemstellung

für die KI explizit und detailreich zu formulieren, oftmals in deutlich klarerer Form als dies für eine manuelle Modellierung durch einen menschlichen Experten erforderlich wäre.

Die Usability des Systems ist grundsätzlich hoch, da der Einstieg über eine einfache Problemformulierung ermöglicht wird. Für fortgeschrittene oder nicht-standardisierte Aufgabenstellungen wird aber auch vom Anwender ein gewisses Maß an Modellierungsverständnis und die Fähigkeit zur Fehlerkontrolle verlangt. Fehler in der Modellstruktur müssen erkannt und gegebenenfalls manuell korrigiert werden. Dies schränkt die vollständige Automatisierung in der Praxis ein.

Repräsentativität der Fallstudien und Übertragbarkeit

Die Auswahl und Umsetzung der Fallstudien verdeutlicht, dass der entwickelte Ansatz für eine breite Palette von klassischen Optimierungsproblemen geeignet ist. Die behandelten Beispiele, von der Personaleinsatzplanung über Produktions- und Verschnittprobleme bis zur Netzwerkoptimierung, bilden typische Aufgaben aus der industriellen und betrieblichen Praxis ab. Dennoch kann nicht ausgeschlossen werden, dass besonders komplexe, unternehmensspezifische oder mehrdeutige Problemstellungen weiterhin manuelle Anpassungen oder eine erweiterte Domänenmodellierung erfordern. Die Übertragbarkeit des Systems auf beliebige Optimierungsprobleme ist damit eingeschränkt.

7. Zusammenfassung und Ausblick

Im abschließenden Rückblick auf diese Arbeit lässt sich feststellen, dass die zentrale Forschungsfrage, inwieweit es mit Hilfe eines KI-basierten Systems auf Basis moderner Large Language Models möglich ist, mathematische Optimierungsmodelle automatisiert aus natürlichsprachlichen Problemstellungen zu generieren und welche technischen, praktischen und qualitativen Anforderungen sowie Grenzen sich bei der Entwicklung und Anwendung eines solchen Ansatzes in praxisorientierten Anwendungsfällen ergeben, umfassend bearbeitet und in zentralen Punkten beantwortet werden konnte. Ausgangspunkt war die Beobachtung, dass mathematische Optimierung als Instrument zur Unterstützung betrieblicher Entscheidungsfindung seit Jahrzehnten ein etabliertes Werkzeug darstellt, dessen praktische Nutzung jedoch häufig durch Hürden wie begrenztes Fachwissen, aufwändige Modellierung und hohe Komplexität eingeschränkt wird. Gleichzeitig hat die Entwicklung leistungsfähiger KI-Methoden, insbesondere im Bereich der Large Language Models, neue Möglichkeiten eröffnet, den Modellierungsprozess zu vereinfachen und einer breiteren Nutzergruppe zugänglich zu machen.

Im Verlauf der Arbeit wurde zunächst der Stand der Wissenschaft und Technik analysiert, um die Grundlagen der mathematischen Optimierung, die Funktionsweise und Einsatzmöglichkeiten von LLMs sowie die klassischen Herausforderungen im Modellierungsprozess transparent herauszuarbeiten. Im Mittelpunkt stand dabei die Erkenntnis, dass ein entscheidender Mehrwert von KI-basierten Systemen in der automatisierten Verarbeitung von natürlichsprachlichen Eingaben liegt, wodurch eine Brücke zwischen Anwendern ohne tiefgehendes Optimierungswissen und den formalen Anforderungen mathematischer Modellierung geschlagen werden kann. Die theoretische und methodische Aufarbeitung unterstrich, dass eine erfolgreiche Modellgenerierung nicht nur technisches Know-how, sondern auch ein Verständnis für die Problemstruktur, für Optimierungsziele und für die unterschiedlichen Modelltypen erfordert.

Ein wesentlicher Schwerpunkt der Arbeit lag auf der Konzeption und prototypischen Umsetzung eines Systems, das unter Verwendung moderner LLMs in der Lage ist, natürlichsprachliche Beschreibungen von Optimierungsproblemen automatisiert in die algebraische Modellierungssprache AMPL zu übersetzen. Im methodischen Vorgehen wurde besonderer Wert auf die klare Trennung von Modellstruktur und Instanzdaten, die systematische Erfassung der Nutzereingaben sowie die Realisierung eines modularen, erweiterbaren Softwaredesigns gelegt. Durch den Aufbau einer intuitiven Benutzerschnittstelle und die Integration leistungsfähiger Solver konnten wichtige Voraussetzungen für den Praxiseinsatz geschaffen werden. Die Herausforderungen im Entwicklungsprozess bestanden unter anderem darin, die vielfältigen Formulierungen und Interpretationsspielräume natürlicher

Sprache so zu adressieren, dass eine möglichst präzise und vollständige Abbildung der Problemstellung im mathematischen Modell erfolgt. Ebenso wurde das System so ausgelegt, dass es flexibel auf unterschiedliche Problemklassen wie lineare, ganzzahlige und gemischt-ganzzahlige Optimierungsmodelle angewendet werden kann.

Im experimentellen Teil der Arbeit wurde das entwickelte System anhand praxisorientierter Fallstudien evaluiert. Hierbei zeigte sich, dass für eine Vielzahl klassischer Optimierungsprobleme aus Bereichen wie der Produktions- und Transportplanung, der Ressourcenallokation oder der Zuordnung von Aufgaben und Kapazitäten formal korrekte und ausführungstaugliche Modelle generiert werden konnten. Die Überprüfung der Modellqualität erfolgte anhand der korrekten Abbildung der Entscheidungsvariablen, Zielfunktionen und Restriktionen, aber auch anhand der Praxistauglichkeit der erzeugten Lösungen im realitätsnahen Anwendungskontext. In der Analyse der Ergebnisse wurde deutlich, dass die Nutzung von LLMs den Modellierungsprozess erheblich beschleunigen und den Zeitaufwand für die initiale Modellbildung deutlich reduzieren kann. Die Anwenderfreundlichkeit und die Reduktion typischer Fehlerquellen wurden in der Evaluation als zentrale Vorteile herausgestellt. Erwähnenswert ist, dass durch die automatisierte Vorgehensweise auch Nutzer ohne tiefgreifende mathematische oder programmiertechnische Vorkenntnisse in die Lage versetzt werden, Optimierungsprobleme zu formulieren und Lösungen zu erarbeiten. Damit trägt die entwickelte Lösung wesentlich zur Demokratisierung mathematischer Optimierung bei und öffnet das Feld für neue Nutzergruppen in Unternehmen und Organisationen.

Allerdings offenbarte die Arbeit auch die Grenzen und Herausforderungen des gewählten Ansatzes. Die Qualität der generierten Modelle ist maßgeblich von der Klarheit und Vollständigkeit der natürlichsprachlichen Eingaben abhängig. Insbesondere bei komplexen, mehrdeutigen oder domänenspezifischen Problemstellungen kam es gelegentlich zu fehlerhaften Modellstrukturen oder unvollständigen Abbildungen, die eine manuelle Nachbearbeitung durch Experten erforderlich machen würden. Es zeigte sich auch, dass gegenwärtige LLMs, trotz ihrer beeindruckenden Fähigkeiten, nicht immer in der Lage sind, tiefes domänenspezifisches Wissen oder implizite Anforderungen korrekt zu erfassen und in mathematische Form zu bringen. Ebenso ist die Nachvollziehbarkeit und Transparenz der automatisiert erzeugten Modelle noch nicht durchgängig gewährleistet, ein Aspekt, der insbesondere für die Akzeptanz und das Vertrauen der Nutzer in der Praxis eine entscheidende Rolle spielt. Daraus ergibt sich die Notwendigkeit, die Modelle und Systeme künftig gezielt weiterzuentwickeln, zu validieren und für spezifische Anwendungsfelder zu optimieren.

Aus diesen Erkenntnissen ergeben sich zahlreiche Implikationen und Ansatzpunkte für die weitere Forschung. Ein vielversprechender Weg liegt in der Entwicklung

dialogorientierter, interaktiver Systeme, die Nutzer nicht nur bei der initialen Modellbildung, sondern auch bei der Überprüfung und Optimierung begleiten und durch Rückkopplung die Modellqualität weiter verbessern. Die gezielte Anpassung und das Finetuning von LLMs auf bestimmte Branchen oder Problemklassen könnten die Domänenkompetenz der Systeme deutlich erhöhen und die praktische Anwendbarkeit weiter steigern. Gleichzeitig müssen begleitende Maßnahmen zur Qualitätssicherung, Validierung und Visualisierung der Modelle entwickelt werden, um die Erklärbarkeit und Transparenz zu stärken. Die Integration von KI-basierten Optimierungssystemen in bestehende betriebliche Entscheidungs- und Planungssysteme eröffnet zudem neue Perspektiven für die Digitalisierung und Automatisierung von Geschäftsprozessen. Allerdings sollten auch die gesellschaftlichen und ethischen Dimensionen, wie etwa Fragen der Verantwortung, Fairness und der Auswirkungen auf Arbeitsprozesse, künftig noch stärker berücksichtigt werden.

Zusammenfassend kann festgehalten werden, dass die automatisierte Generierung mathematischer Optimierungsmodelle mit Hilfe moderner Large Language Models ein zukunftsweisendes Feld darstellt, das in Wissenschaft und Praxis zunehmend an Bedeutung gewinnt. Die Arbeit hat gezeigt, dass technische Umsetzbarkeit, Praxistauglichkeit und Qualität KI-basierter Modellgenerierung bereits heute in vielen Anwendungsfällen gegeben sind, dass aber auch zentrale Herausforderungen bestehen, die weiterer Forschung und Entwicklung bedürfen. Die Ergebnisse verdeutlichen, dass ein interdisziplinärer Ansatz, der das Potenzial der Künstlichen Intelligenz, das methodische Wissen aus dem Operations Research und die Anforderungen der Praxis vereint, die besten Voraussetzungen bietet, um die Potenziale der automatisierten Optimierung weiter auszuschöpfen und langfristig tragfähige, effiziente und vertrauenswürdige Systeme zu schaffen. Letztlich bleibt festzuhalten, dass die Demokratisierung der mathematischen Optimierung durch KI nur gelingen kann, wenn neben der technischen Weiterentwicklung auch die Integration in betriebliche Prozesse und die Berücksichtigung gesellschaftlicher Anforderungen konsequent vorangetrieben werden.

Tabellenverzeichnis

Abkürzungsverzeichnis

KI	Künstliche Intelligenz
LLM	Large Language Model(s)
AMPL	A Mathematical Programming Language
DSR	Design Science Research
GPT	Generative Pre-trained Transformer (z. B. GPT-3, GPT-4o)

Literaturverzeichnis

Abosaooda, H. N., Ariffin, S. B., Mohammed Alyasiri, O., & Noor, A. A. (2025). Evaluating the Effectiveness of AI Tools in Mathematical Modelling of Various Life Phenomena: A Proposed Approach. InfoTech Spectrum: Iraqi Journal of Data Science, 2(1), 16–25. https://doi.org/10.51173/ijds.v2i1.16

Biermann-Wehmeyer, N. (2024). Die Ursprünge der Digitalisierung – Eine detaillierte Betrachtung ihrer Entstehung & Entwicklung!, https://bildungsinstitut-wirtschaft.de/die-ursprueenge-der-digitalisierung/, Abruf am 11.07.2025.

Blank, I. A. (2023). What are large language models supposed to model? Trends in Cognitive Sciences, 27(11), 987–989. https://doi.org/10.1016/j.tics.2023.08.006

Bosch, R., & Trick, M. (2014). Integer Programming. In Burke, E. K., & Kendall, G. (Hrsg.), Search Methodologies: Introductory Tutorials in Optimization and Decision Support Techniques (67–92). Springer US. https://doi.org/10.1007/978-1-4614-6940-7_3

Brown, T., Mann, B., Ryder, N., Subbiah, M., Kaplan, J. D., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., & Askell, A. (2020). Language models are few-shot learners. Advances in Neural Information Processing Systems, 33, 1877–1901.

Buscema, M. (1998). Theory: Foundations of Artificial Neural Networks. Substance Use & Misuse, 33(1), 17–199. https://doi.org/10.3109/10826089809115858

Buscema, M. (2002). A brief overview and introduction to artificial neural networks. Substance Use & Misuse, 37(8–10), 1093–1148. https://doi.org/10.1081/JA-120004171

Bynum, M. L., Hackebeil, G. A., Hart, W. E., Laird, C. D., Nicholson, B. L., Siirola, J. D., Watson, J.-P., & Woodruff, D. L. (2021). Pyomo – Optimization Modeling in Python (Bd. 67). Springer.

Chaudhary, H., Sharma, G., Nishad, D. K., & Khalid, S. (2025). Advanced queueing and scheduling techniques in cloud computing using AI-based model order reduction. Discover Computing, 28(1), 75. https://doi.org/10.1007/s10791-025-09581-7

Devlin, J., Chang, M.-W., Lee, K., & Toutanova, K. (2019). Bert: Pre-training of deep bidirectional transformers for language understanding. Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers).

Dogru, A. K., & Keskin, B. B. (2020). AI in operations management: applications, challenges and opportunities. Journal of Data, Information and Management, 2(2), 67–74. https://doi.org/10.1007/s42488-020-00023-1

Doshi-Velez, F., & Kim, B. (2017). Towards a rigorous science of interpretable machine learning. arXiv Preprint arXiv:1702.08608.

Dubey, R., Gunasekaran, A., & Papadopoulos, T. (2024). Benchmarking operations and supply chain management practices using Generative AI: Towards a theoretical framework. Transportation Research Part E: Logistics and Transportation Review, 189, 103689. https://doi.org/10.1016/j.tre.2024.103689

Fatawi, I., Asy'ari, M., Hunaepi, H., Samsuri, T., & Bilad, M. R. (2024). Empowering language models through advanced prompt engineering: A comprehensive bibliometric review. Indonesian Journal of Science and Technology, 9(2), 441–462.

Fosso Wamba, S., Cameron, G. M., Q. M., & Minner, S. (2024). ChatGPT and generative artificial intelligence: an exploratory study of key benefits and challenges in operations and supply chain management. International Journal of Production Research, 62(16), 5676–5696. https://doi.org/10.1080/00207543.2023.2294116

Fosso Wamba, S., Queiroz, M. M., Chiappetta Jabbour, C. J., & Shi, C. (2023). Are both generative AI and ChatGPT game changers for 21st-Century operations and supply chain excellence? International Journal of Production Economics, 265, 109015. https://doi.org/10.1016/j.ijpe.2023.109015

Fourer, R., Gay, D. M., & Kernighan, B. W. (2003). AMPL: A mathematical programming language. Management Science, 36(5), 519–554.

Fraunhofer-Institut für Integrierte Schaltungen IIS, Bereich SCS. Mathematische Optimierung, https://www.scs.fraunhofer.de/de/referenzen/ada-center/mathematische-optimierung.html, Abruf am 21.07.2025.

Ganthavee, V., Fernando, M. M. R., & Trzcinski, A. P. (2024). Monte Carlo Simulation, Artificial Intelligence and Machine Learning-based Modelling and Optimization of Three-dimensional Electrochemical Treatment of Xenobiotic Dye Wastewater. Environmental Processes, 11(3), 41. https://doi.org/10.1007/s40710-024-00719-1

Gay, D. M. (2015). The AMPL modeling language: An aid to formulating and solving optimization problems. In Numerical Analysis and Optimization: NAO-III, Muscat, Oman, Januar 2014 (95–116). Springer.

Google. (2025a). About OR-Tools, https://developers.google.com/optimization, Abruf am 12.07.2025.

Google. (2025b). Introduction, https://developers.google.com/optimization/introduction, Abruf am 12.07.2025.

Hestness, J., Narang, S., Ardalani, N., Diamos, G., Jun, H., Kianinejad, H., Patwary, M. M. A., Yang, Y., & Zhou, Y. (2017). Deep learning scaling is predictable, empirically. arXiv Preprint arXiv:1712.00409.

Hevner, A., & Chatterjee, S. (2010). Design Science Research in Information Systems. In Hevner, A., & Chatterjee, S. (Hrsg.), Design Research in Information Systems: Theory and Practice (9–22). Springer US.

Hevner, A. R., March, S. T., Park, J., & Ram, S. (2004). Design Science in Information Systems Research. MIS Quarterly, 28(1), 75–105. https://doi.org/10.2307/25148625

Hillier, F. S., & Liebermann, G. J. (2014). Operations Research: Einführung. Walter de Gruyter GmbH & Co KG.

Jusevičius, V., Oberdieck, R., & Paulavičius, R. (2021). Experimental analysis of algebraic modelling languages for mathematical optimization. Informatica, 32(2), 283–304. https://doi.org/10.15388/21-infor447

Kadıoğlu, S., Pravin Dakle, P., Uppuluri, K., Politi, R., Raghavan, P., Rallabandi, S., & Srinivasamurthy, R. (2024). Ner4Opt: Named entity recognition for optimization modelling from natural language. Constraints, 29(3), 261–299. https://doi.org/10.1007/s10601-024-09376-5

Karbowski, A., & Wyskiel, K. (2021). Comparative study of AMPL, Pyomo and JuMP optimization modeling languages on a network linear programming problem example. Pomiary Automatyka Robotyka, 25.

Kelbert, P., Siebert, J., & Jöckel, L. (2023, 12.12.2023). Was sind Large Language Models? Und was ist bei der Nutzung von KI-Sprachmodellen zu beachten? Blog des Fraunhofer-Instituts für Experimentelles Software Engineering IESE, https://www.iese.fraunhofer.de/blog/large-language-models-ki-sprachmodelle/, Abruf am 21.07.2025.

Koop, A., & Moock, H. (2023). Lineare Optimierung – eine anwendungsorientierte Einführung in Operations Research (3. Aufl.). Springer Spektrum Berlin, Heidelberg. https://doi.org/10.1007/978-3-662-66387-5

Liu, Y., Han, T., Ma, S., Zhang, J., Yang, Y., Tian, J., He, H., Li, A., He, M., Liu, Z., Wu, Z., Zhao, L., Zhu, D., Li, X., Qiang, N., Shen, D., Liu, T., & Ge, B. (2023). Summary of ChatGPT-related research and perspective towards the future of large language models. Meta-Radiology, 1(2), 100017. https://doi.org/10.1016/j.metrad.2023.100017

Luo, C., Li, A.-J., Xiao, J., Li, M., & Li, Y. (2024). Explainable and generalizable AI-driven multiscale informatics for dynamic system modelling. Scientific Reports, 14(1), 18219. https://doi.org/10.1038/s41598-024-67259-4

Luzzi, M., Guerriero, F., Maratea, M., Greco, G., & Garofalo, M. (2025). ChatGPT and operations research: Evaluation on the shortest path problem. Soft Computing, 29(3), 1407–1418. https://doi.org/10.1007/s00500-025-10505-2

Mellouli, T. (2015). Optimierung, Netzwerke und Transportlogistik, WS 14/15. Vorlesungsskript, Martin-Luther-Universität Halle-Wittenberg.

Mellouli, T. (2021). Optimierung, Netzwerke und Transportlogistik – Fallstudien und Software, WS 20/21. Vorlesungsskript, Martin-Luther-Universität Halle-Wittenberg.

Min, B., Ross, H., Sulem, E., Pouran Ben Veyseh, A., Huu Nguyen, T., Sainz, O., Aguirre, E., Heintz, I., & Roth, D. (2023). Recent advances in natural language processing via large pre-trained language models: A survey. ACM Computing Surveys, 56(2), Artikel 30. https://doi.org/10.1145/3605943

MiniZinc Team. (2022). The MiniZinc Handbook 2.6.4: Introduction, https://docs.minizinc.dev/en/2.6.4/intro.html, Abruf am 21.07.2025.

Nishad, D. K., Tiwari, A. N., Khalid, S., Gupta, S., & Shukla, A. (2024). AI based UPQC control technique for power quality optimization of railway transportation systems. Scientific Reports, 14(1), 17935. https://doi.org/10.1038/s41598-024-68575-5

Obinwanne, U., & Feng, W. (2025). A case study on AI to automate simulation modelling. Software and Data Engineering, Cham.

Ramamonjison, R., Yu, T., Li, R., Li, H., Carenini, G., Ghaddar, B., He, S., Mostajabdaveh, M., Banitalebi-Dehkordi, A., Zhou, Z., & Zhang, Y. (2022). NL4Opt Competition: Formulating optimization problems based on their natural language descriptions. In

Proceedings of the NeurIPS 2022 Competitions Track, Proceedings of Machine Learning Research. https://proceedings.mlr.press/v220/ramamonjison23a.html

Schmidhuber, J. (2015). Deep learning in neural networks: An overview. Neural Networks, 61, 85–117. https://doi.org/10.1016/j.neunet.2014.09.003

Shu, X. (2020). Artificial neural networks. In Knowledge Discovery in the Social Sciences (1. Aufl., 191–206). University of California Press. https://doi.org/10.2307/j.ctvw1d683.12

Sudermann-Merx, N. (2023). Optimierungsmodelle in der Praxis. In Sudermann-Merx, N. (Hrsg.), Einführung in Optimierungsmodelle: Mit Beispielen und Real-World-Anwendungen in Python (195–202). Springer Berlin Heidelberg. https://doi.org/10.1007/978-3-662-67381-2_8

Sundberg, L., & Holmström, J. (2023). Democratizing artificial intelligence: How no-code AI can leverage machine learning operations. Business Horizons, 66(6), 777–788. https://doi.org/10.1016/j.bushor.2023.04.003

Wang, L., & Zhao, J. (2023). Architecture of Advanced Numerical Analysis Systems: Designing a Scientific Computing System Using OCaml. Springer Nature.

Williams, H. P. (2013). Model building in mathematical programming. John Wiley & Sons.

Wulff, P., Kubsch, M., & Krist, C. (2025). Basics of machine learning. In Wulff, P., Kubsch, M., & Krist, C. (Hrsg.), Applying Machine Learning in Science Education Research: When, How, and Why? (15–48). Springer Nature Switzerland. https://doi.org/10.1007/978-3-031-74227-9_2

Yu, W., Zhu, C., Li, Z., Hu, Z., Wang, Q., Ji, H., & Jiang, M. (2022). A survey of knowledge-enhanced text generation. ACM Computing Surveys, 54(11s), Artikel 227. https://doi.org/10.1145/3512467

Yuksekgonul, M., Bianchi, F., Boen, J., Liu, S., Lu, P., Huang, Z., Guestrin, C., & Zou, J. (2025). Optimizing generative AI by backpropagating language model feedback. Nature, 639(8055), 609–616. https://doi.org/10.1038/s41586-025-08661-4

Anhang

Der Quellcode zu dieser Arbeit ist unter folgendem Link verfügbar:

https://github.com/wjens97/Masterarbeit_Jensen

Python Code des Modells:

```
from amplpy import AMPL, modules
modules.install()
ampl = AMPL()
import openai
import subprocess
import tempfile
import os
import re
import time
# 1. OpenAI API-Key
openai.api_key = "hier API-Key einfügen"
# 2. Optimierungs-Aufgabe als Freitext
user_problem = """
Aufgabe hier einfügen
"""

# 3. GPT Prompt – KI soll AMPLpy-Kompatiblen Code liefern
gpt_prompt = f"""
Formuliere folgendes Optimierungsproblem als lauffähigen Python-Code mit
amplpy:
- Das AMPL-Modell enthält ausschließlich Variablen, Zielfunktion und Neben-
bedingungen.
- Mengen und Parameter WERDEN im AMPL-Modell **definiert** (z.B. 'set PRO-
DUCTS;'), aber NICHT mit Daten befüllt!
- ALLE Daten (Mengen/Parameter/Werte) werden AUSSCHLIESSLICH im Python-Teil
per ampl.set[...] und ampl.param[...] gesetzt.
- KEINE AMPL-Funktionen wie ord(), prev(), next(), first(), last() verwen-
den.
- Setze die Lagerbilanz explizit für jede Periode, z.B. durch einzelne Cons-
traints für jede Periode.
- Initialisiere amplpy so: from amplpy import AMPL, modules; modules.in-
stall(); ampl = AMPL()
- Die Entscheidungsvariablen im Modell sollen als ganzzahlig (integer) de-
klariert werden, z.B.: `var x {{A, S}} integer >= 0;`.
- Nach dem Lösen sollen die Variablenwerte und das Ziel im Terminal ausgege-
ben werden.
- Die Ausgabe aller Variablenwerte erfolgt durch Schleifen über die Index-
mengen im Python-Teil, z.B. for i in ampl.getSet('...'): for j in ampl.get-
Set('...'): print(...).
- Die Entscheidungsvariablen im Modell sollen als ganzzahlig (integer) de-
klariert werden, z.B.: var x {{A, S}} integer >= 0; .
```

```
- Verwende im Python-Teil vor ampl.solve() die Zeile ampl.setOption('sol-
ver', 'highs') oder ampl.setOption('solver', 'cbc').
- **Füge im Python-Code vor und nach ampl.solve() eine Zeitmessung mit time
ein und gib die Solver-Laufzeit nach dem Lösen im Terminal aus.**
{user_problem}
"""
# 4. GPT-Aufruf
start_gen = time.time()
client = openai.OpenAI(api_key=openai.api_key)
response = client.chat.completions.create(
    model="gpt-4o",
    messages=[{"role": "user", "content": gpt_prompt}],
    temperature=0,
)
end_gen = time.time()
print(f"Laufzeit Modellgenerierung: {end_gen - start_gen:.2f} Sekunden")

content = response.choices[0].message.content
code_match = re.search(r"```python(.*?)```", content, re.DOTALL)
code = code_match.group(1).strip() if code_match else ""

# 5. Problematische AMPL-Konstrukte filtern
def is_ampl_code_problematic(code):
    forbidden = [
        'data;', ':=', 'set ',  # keine Mengen-/Parameterzuweisung im Modell
selbst
        'ord(', 'prev(', 'next(', 'first(', 'last(',
        'ampl.eval("set', 'ampl.eval(\'set'
    ]
    # Ausnahme: set ...; ist im Modell ok, set ... := ... nicht!
    return any(x in code for x in forbidden if not x == 'set ' or ':=' in
code or 'set ' in code and ':=' in code)

# 6. UTF-8-Encoding-Header
header = "# -*- coding: utf-8 -*-\n"
code = header + code
# 7. Speichern & Ausführen
with tempfile.NamedTemporaryFile("w", delete=False, suffix=".py", encod-
ing="utf-8") as f:
    f.write(code)
    filename = f.name
print(f"Generierter Code gespeichert unter: {filename}")
print("Starte Ausführung...\n---\n")
subprocess.run(["python", filename])
```

Fallstudie „Krankenhaus – Optimale Pflegerzuordnung“

Ein Krankenhaus hat 3 Abteilungen: Abt. I-Intensivstation, Abt. II-Orthopädie, Abt. III-Entbindungsstation. Es gibt 3 Arbeitsschichten im Krankenhaus, die verschiedene Anforderungen an benötigten

Pfleger(innen) haben. Das Krankenhaus möchte die minimale Anzahl insgesamt benötigter Pfleger bestimmen, wobei folgende Bedingungen erfüllt werden müssen:

1) Das Krankenhaus muss mindestens 13, 32 und 22 Pfleger zu den Abteilungen I, II bzw. III (über alle Schichten) zuordnen.

2) Das Krankenhaus muss mindestens 26, 24 und 19 Pfleger zu den Schichten 1, 2 bzw. 3 (über alle Abteilungen) zuordnen.

3) Die minimal und maximal benötigte Anzahl von Pfleger, die jeweils zu einer bestimmten Abteilung und zu einer bestimmten Schicht zugeordnet werden müssen, sind in folgender Tabelle angegeben:

Schicht	**Abt. I**	**Abt. II**	**Abt. III**
1	6, 8	11, 12	7, 12
2	4, 6	11, 12	7, 12
3	2, 4	10, 12	5, 7

Formulieren Sie dieses Problem als Netzwerkflussmodell und lösen Sie es mit geeigneten Methoden aus

der Vorlesung.

Fallstudie „Verschnittproblem – Papierfabrik“

Eine Papierfabrik erhielt drei Aufträge für Papierrollen mit festgelegten Breiten und Längen wie in der folgenden Tabelle angegeben:

Auftrag	**Format: Breite (in m)**	**Geforderte Länge (in m)**
A	5	15.000
B	7	35.000
C	9	20.000

Die Fabrik produziert Papierrollen in 2 Standardbreiten: 20 m (Typ 1) und 10 m (Typ 2), die je nach geforderten Breiten in den Aufträgen zurecht geschnitten werden, und zwar Breiten zu Breiten und nicht zu Längen:

Man kann nicht etwa eine Typ A-Rolle in Längen von z.B. 7 m schneiden, dann den Auftrag B durch Zusammenkleben von 5.000 solcher Teile erfüllen.

Beachte jedoch, dass für einen Auftrag die geforderte Länge aus mehreren Teilen (ihrer Längen nach) zu einer Rolle „zusammengeklebt“ werden kann. D.h. man kann Auftrag B durch Benutzung von 10.000 m von Typ 1 Rolle und 15.000 m von Typ 2 Rolle erfüllen, nämlich Typ 1 Rolle liefert 2 mal Breite 7m, also 2 Breite-7m-Teile jeweils der Länge 10.000 m und Typ 2 Rolle liefert 1 mal Breite 7m (ein Teil der Länge 15.000m). Durch Zusammenkleben der 3 Teile bekommt man die geforderte Länge von 35.000m der Breite 7m.

Verschnittproblem: Bestimmen Sie eine Schnittstrategie der Standardrollen vom Typ 1 und 2, die alle Aufträge erfüllt, mit der Zielsetzung (1), den insgesamt entstehenden Verschnitt (Menge der unbrauchbaren Papierteile in m2) zu minimieren.

1. Nehmen Sie an, dass die Breiten in den Aufträgen A, B und C oft von Kunden angefordert werden; d.h. Mehrproduktion von diesen Breiten schadet der Firma nicht. Formulieren Sie ein LP-Modell für das oben definierte Verschnittproblem, wobei mindestens die in den Aufträgen geforderten Längen jeweils von den verschiedenen Breiten produziert werden.

2. Nehmen Sie nun an, dass die Firma die Mehrproduktion von den Breiten in den Aufträgen A, B und C nicht einfach absetzen kann, d.h. Mehrproduktion von diesen Breiten wird als Verschnitt angesehen. Formulieren Sie nun das Verschnittproblem unter diesen veränderten Voraussetzungen.

3. Die Kosten für das verwendete Papier sind bekannt, z.B. 1m von Typ 1 Rolle kostet 0,18 DM und 1m von Typ 2 Rolle kostet 0,1 DM). Bearbeiten Sie Teilaufgaben 1 und 2 unter der Zielsetzung (2), die Gesamtkosten für das verwendete Papier zu minimieren.

4. Löse diese LP-Modelle mit Hilfe einer Optimierungssoftware.

Die verwendeten user_problem:

Personaleinsatzplanung im Krankenhaus:

user_problem = """

Ein Krankenhaus hat 3 Abteilungen (I: Intensivstation, II: Orthopädie, III: Entbindungsstation) und 3 Arbeitsschichten. Zu jeder Abteilung und Schicht müssen bestimmte Mindest- und Maximalzahlen an Pflegern eingeteilt werden (siehe Tabelle).

Gesamtbedarf (über alle Schichten/Abteilungen): I: mind. 13, II: mind. 32, III: mind. 22 Pfleger.

In den Schichten 1, 2, 3 müssen insgesamt mind. 26, 24, 19 Pfleger eingeteilt werden.

Schicht/Abt	I	II	III
1	6-8	11-12	7-12
2	4-6	11-12	7-12
3	2-4	10-12	5-7

Stelle ein Optimierungsmodell auf, das die minimale **Gesamtanzahl** an Pflegern bestimmt, sodass alle Bedingungen eingehalten werden.

"""

OP-Planung in einer Klinik:

user_problem = """

In einer Klinik sollen in einer Woche verschiedene Operationen (Herz, Orthopädie, Viszeral, Unfallchirurgie, HNO) geplant werden.

Für jede Operation sind bestimmte OP-Zeiten (in Stunden) und eine Mindestanzahl von Pflegenden und Ärzten notwendig.

Die Klinik verfügt über 4 OP-Säle, 12 OP-Tage (4 Säle × 3 Tage), 60 verfügbare OP-Stunden pro Woche, 15 Pflegekräfte und 10 Ärzte.

Es existieren folgende OP-Anforderungen und Gewinne:

OP-Art	Gewinn (Euro)	OP-Zeit (h)	Pflegebedarf	Ärztebedarf	max. Patienten
Herz	5000	6	4	3	4
Orthopädie	3500	4	3	2	8
Viszeral	4200	5	3	2	6
Unfallchirurgie	2500	3	2	1	10
HNO	1800	2	1	1	8

Wie viele Operationen jeder Art sollen geplant werden, um den Gesamtgewinn zu maximieren?

Berücksichtige alle Kapazitäts- und Personalrestriktionen.

"""

Verschnittoptimierung bei Papierrollen:

user_problem = """

Eine Papierfabrik hat drei Kundenaufträge für Papierrollen:

- Auftrag A: 5 m Breite, 15.000 m Länge

- Auftrag B: 7 m Breite, 35.000 m Länge

- Auftrag C: 9 m Breite, 20.000 m Länge

Die Fabrik produziert Papierrollen in zwei Standardbreiten:

- Rollen-Typ 1: 20 m breit

- Rollen-Typ 2: 10 m breit

Jeder Auftrag darf in beliebig viele Teilstücke geschnitten werden (eine Klebung in Längsrichtung ist erlaubt), solange für jeden Auftrag die bestellte Gesamtlänge und -breite vollständig geliefert wird.

Optimierungsziel:

Bestimmen Sie für jeden Auftrag, wie viele Laufmeter aus jeder Rollenbreite zugeschnitten werden müssen, sodass:

- Für jeden Auftrag mindestens die geforderte Gesamtfläche (Breite × Länge) geliefert wird.

Das heißt: Die Summe der zugeschnittenen Länge aus allen Rollenbreiten, multipliziert mit der jeweiligen Rollenbreite, muss mindestens der geforderten Fläche des Auftrags entsprechen.

(Formal: ∑ [zugeschnittene Länge × Rollenbreite] ≥ Auftragsbreite × Auftragslänge)

- Der gesamte Verschnitt (also die Summe der unbrauchbaren Papierfläche über alle Aufträge, berechnet als (Rollenbreite – Auftragsbreite) × zugeschnittene Meter) wird minimiert.

Gesucht sind daher die optimalen Zuschnittlängen (in Metern) aus jeder Rollenbreite für jeden Auftrag, sodass die Flächenanforderungen erfüllt und der Gesamtverschnitt minimiert wird.

"""

Mehrperiodige Produktionsplanung mit Wartung:

user_problem = """

Eine Firma produziert drei Produkte (P1, P2, P3) auf zwei Maschinen (M1, M2) über drei Monate.

• Deckungsbeiträge (Euro/Stück): P1: 10, P2: 8, P3: 7

• Zeitbedarf pro Produkt/Maschine (in Stunden):

M1: P1: 1, P2: 2, P3: 1

M2: P1: 2, P2: 1, P3: 2

• Maschinenkapazität pro Monat: M1: 60 Std, M2: 50 Std.

• Wartung: Im zweiten Monat steht M2 nur 30 Std zur Verfügung.

• Nachfrage pro Monat: siehe Tabelle.

| | Monat 1 | Monat 2 | Monat 3 |

|------|---------|---------|---------|

| P1 | 10 | 12 | 8 |

| P2 | 15 | 10 | 12 |

| P3 | 7 | 8 | 9 |

• Es gibt keinen Anfangslagerbestand. Am Ende von Monat 3 sollen mindestens 3 Stück von jedem Produkt auf Lager sein.

• Die Lagerkapazität ist 10 Stück pro Produkt/Monat (Lagerkosten: 1 €/Stück/Monat).

Frage: Wie viele Stück von jedem Produkt sollen pro Monat produziert und verkauft werden, um den Gesamtgewinn abzüglich Lagerkosten zu maximieren? Berücksichtige alle Maschinen-, Nachfrage- und Wartungsrestriktionen.

"""

Netzwerkoptimierung: Flughafen-Logistik:

user_problem = """

Ein internationaler Flughafen möchte die optimale Versorgung mehrerer Terminals mit Lebensmitteln und Duty-Free-Artikeln planen. Es gibt drei zentrale Warenlager (Nord, Süd, Ost) und vier Terminals (T1–T4).

Jedes Lager hat eine maximale Versandkapazität pro Tag:

- Nord: 100 Paletten

- Süd: 80 Paletten

- Ost: 70 Paletten

Jedes Terminal benötigt eine Mindestmenge an Waren pro Tag:

- T1: 60 Paletten

- T2: 50 Paletten

- T3: 70 Paletten

- T4: 40 Paletten

Die Transportmöglichkeiten sind wie folgt (Kapazität pro Tag, keine Direktverbindung = Kapazität 0):

- Nord → T1: 60, Nord → T3: 40

- Süd → T1: 30, Süd → T2: 50, Süd → T4: 30

- Ost → T2: 30, Ost → T3: 40, Ost → T4: 30

Jedes Terminal kann von mehreren Lagern beliefert werden, aber die Summe aller Lieferungen je Lager bzw. Terminal darf deren jeweilige Kapazitätsgrenzen nicht überschreiten.

Aufgabe:

Bestimmen Sie, wie viele Paletten täglich von jedem Lager zu jedem Terminal transportiert werden sollen, sodass alle Terminals vollständig versorgt sind und keine Kapazitätsgrenze (weder beim Versand noch beim Empfang) überschritten wird. Das Ziel ist es, den gesamten Warentransport zu minimieren (Summe aller gelieferten Paletten).

Ergänzung für die Implementierung:

Für jede mögliche Kombination aus Lager (Nord, Süd, Ost) und Terminal (T1–T4), für die **keine Transportmöglichkeit besteht**, soll die Transportkapazität explizit auf **0** gesetzt werden. Die

vollständige Kapazitätsmatrix muss im Modell abgebildet werden, auch für Routen, die im Netzwerk nicht existieren.

"""

Erklärung

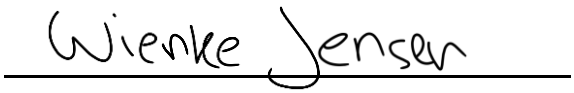
Ich versichere, dass die vorliegende Masterarbeit von mir selbstständig und ohne Benutzung anderer als der angegebenen Hilfsmittel angefertigt wurde. Alle Stellen, die wörtlich oder sinngemäß aus veröffentlichten und nicht veröffentlichten Schriften entnommen wurden, sind als solche kenntlich gemacht.

Diese Arbeit ist in gleicher oder ähnlicher Form noch nicht als Prüfungsarbeit eingereicht worden.

Ich versichere zudem, dass ich keine Master-Prüfung in einem wirtschaftswissenschaftlichen Studiengang an einer Hochschule oder eine gleichwertig angerechnete Prüfung endgültig nicht bestanden habe.

Ich versichere weiterhin, dass ich meinen Prüfungsanspruch nicht durch Versäumen einer Wiederholungsfrist verloren habe und dass ich mich nicht in einem schwebenden Verfahren zur Master-Prüfung oder einer vergleichbaren Prüfung für einen wirtschaftswissenschaftlichen Studiengang an einer anderen Hochschule befinde.

Ich wurde darüber belehrt, dass die vorliegende Arbeit mit Null Punkten als nicht bestanden bewertet wird, wenn die vorstehende Erklärung unrichtig oder unvollständig ist.



Wienke Jensen Halle (Saale), 22.07.2025