Institut für Mathematik
Fakultät II – Mathematik und Naturwissenschaften
Technische Universität Berlin

# Praktikumsbericht

Hannes Tarun
357632
Technomathematik

Betreuer / Betreuerin: Lars Stolletz
Institution / Unternehmen: 4flow AG

01.12.2023

# Inhaltsverzeichnis

**1 Vorwort** **3**

**2 Das Unternehmen** **3**
2.1 Daten und Fakten . . . . . 3
2.2 Konzept und Unternehmensphilosophie . . . . . 3

**3 Beschreibung meiner Tätigkeiten und Erläuterung der Resultate** **4**
3.1 Allgemeiner Tätigkeitsbereich . . . . . 4
3.2 Hauptversorgeroptimierung . . . . . 5
3.3 Truckloadoptimierung . . . . . 7

**4 Fazit und Ausblick** **8**

# 1 Vorwort

Die allgemeine Studien- und Prüfungsordnung für den Bachelorstudiengang Technomathematik ([Ber22]) sieht die Absolvierung eines Praktikums vor, während dessen *„ein verstärkter Bezug zu der Tätigkeit und den Aufgaben eines Mathematikers/einer Mathematikerin in der Praxis hergestellt“* werden soll.
Die **4flow AG** ermöglichte mir den Einblick in eben diese Beschäftigungsfelder nicht nur im Rahmen eines Praktikums, sondern bot mir im Oktober 2016 eine Stelle als studentischer Mitarbeiter an. In den seitdem vergangenen 7 Jahren habe ich erst als Werksstudent und dann (seit 2020) in Teilzeitbeschäftigung (24h/Woche) sehr umfangreich Erfahrungen über den Berufsalltag und die Anforderungen an Mathematiker:innen im Berufsalltag sammeln können. Meine Beschäftigung dauert bis zum heutigen Tage an und ich freue mich darauf, auch nach Abschluss meines Studiums weitere Berufserfahrung bei diesem Unternehmen gewinnen zu können.

Durch die große Dauer meiner Berufstätigkeit für die 4flow AG ist die Liste meiner Aufgabenfelder und Arbeitsleistungen lang. Die in Kapitel 3 aufgelistet Tätigkeiten stellen daher nur eine kurze Stichprobe des bisher geleisteten dar.

# 2 Das Unternehmen

## 2.1 Daten und Fakten

Die **4flow AG** ist ein im Jahr 2000 in Berlin gegründetes Unternehmen im Bereich der Logistikdienstleistung. Von den damals circa 400 Arbeitnehmer:innen, die 4flow zu Beginn meiner Tätigkeit im Jahr 2016 beschäftigte, wuchs das Team auf heute[1] über 1000 Mitarbeitende. Diese sind neben dem Hauptsitz in Berlin Charlottenburg an über 20 weiteren Standorten nicht nur in Deutschland und Europa[2] tätig, sondern auch in Asien, Nord- und Südamerika. Dabei arbeitet 4flow für und mit Firmen aus vielen verschiedenen Branchen, vor allem mit Unternehmen aus dem Bereich der Fahrzeughersteller und -zulieferer sowie des Maschinen- und Anlagenbaus.

## 2.2 Konzept und Unternehmensphilosophie

Das Unternehmensmodell steht auf drei Säulen:

- **Logistikberatung** - hier werden die Kunden von 4flow dabei unterstützt und begleitet, ihre aktuelle Supply Chain und ihre Logistikprozesse zu analysieren, auf Optimierungspotential zu untersuchen und strategische Entscheidungen (bspw. Eröffnung eines neuen Lagerstandorts) zu treffen und umzusetzen. In zeitlich begrenzten Beratungsprojekten betrachtet *4flow consulting* dabei in enger Zusammenarbeit mit den Auftraggebern viele Facetten der Liefer- und Logistiknetzwerke, z.B. Kosten, Nachhaltigkeit, Robustheit und Effizienz.

- **Logistikmanagment** - 4flow bietet eine 4PL-Dienstleistung[3], d.h. 4flow steuert und/oder übernimmt die Koordination von täglichen, operativen Prozessen und Aufgaben in der Supply Chain, ohne dabei selbst Sachwerte wie Fahrzeuge oder Lagerhallen zu besitzen. *4flow management* reagiert dabei auf die kurzfristigen und wechselnden Anforderungen an Kundennetzwerke und -transporte und optimiert eben diese kontinuierlich. (bspw. Bestellung eines weiteren Fahrzeugs, Umroutung von Transporten bei Zeitverzögerungen o.ä.)

- **Logistiksoftware** - *4flow software* stellt seinen Kunden verschiedene Software-Produkte zur Verfügung, die bei der Visualisierung, Analyse, Optimierung und Verwaltung verschiedener Teile der Supply Chain unterstützen. Dabei umfasst das Portfolio sowohl alleinstehende Komponenten, als auch Möglichkeiten zur Integration in bereits bestehende Software-Systeme beim Kunden. Damit können Problemstellungen in den verschiedenen Teilen der Produktions- und Transportnetzwerke bearbeitet werden - von der Materialverfügbarkeits- und Lagerplanung bis hin zur 3D-Packplanung einzelner Transporte.

Desweiteren unterhält 4flow weitere Abteilungen, beispielsweise ein Data-Science Team und die eigene Forschungsabteilung *4flow research*, die in enger Zusammenarbeit mit Universitäten, Instituten und Unternehmen Forschungsprojekte[4] durchführt und in Studien und Pilotprojekten neue Methoden und Innovationen in der Logistik evaluiert.

Wesentliches Merkmal der Firmenphilosophie ist die enge Verzahnung der verschiedenen Unternehmensbereiche. Die genannten Abteilungen agieren nicht isoliert, sondern stehen in regem Austausch miteinander. So werden unter anderem die kunden- und industriespezifischen Erkenntnisse aus einmaligen Beratungsaufträgen abstrahiert und für den regelmäßigen Optimierungsbetrieb im Logistigmanagement nutzbar gemacht. Weiteres Beispiel der Kooperation ist die Nutzung der von 4flow entwickelten Software in den anderen Businesslines. Sowohl bei den strategischen und langfristigen Entscheidungen, die *4flow consulting* treffen muss, als auch bei der tagesaktuellen Transportplanung und -optimierung im Bereich *4flow management*, sind die Programme und Tools in regelmäßigem Einsatz.

[1] Stand Oktober 2023 (siehe auch [4fl23])
[2] bspw. Frankreich, Spanien, Ungarn, ... (siehe auch [4fl23])
[3] i.e. *fourth party logistics*
[4] siehe u.a. `https://www.4flow.de/unternehmen/forschung-und-innovation/forschungsprojekte.html`

Dieser ganzheitliche Ansatz ermöglicht es 4flow, seine Kunden von der Analyse ihrer Logistik-Netzwerke, über deren Konzeption und Optimierung, bis hin zur Umsetzung zu begleiten.

# 3 Beschreibung meiner Tätigkeiten und Erläuterung der Resultate

## 3.1 Allgemeiner Tätigkeitsbereich

Ich bin bei 4flow in der Businessline *4flow software* als **Software Developer** tätig. Schon sehr früh habe ich gelernt, wie viele verschiedene Aufgaben dieser Beruf umfasst. Dazu zählen

- Software-Entwicklung - das Übersetzen von technischen Anforderungen in Code. Dies umfasst sowohl die vollständige *Neuentwicklung* von neuen Features, bspw. neuen Algorithmen, als auch die Erweiterung bestehender Codebestandteile, z.B. das Einfügen neuer Restriktionen. Das erfordert bspw. die Auswahl richtiger Datenstrukturen und einer geeigneten Softwarearchitektur. Anderer wesentlicher Bestandteil ist das *Bugfixing* - das Beheben von Defekten im Code. Dafür muss der fehlerhafte Programmteil identifiziert, analysiert, auf sein Verhalten in Rand- und Extremfällen untersucht und das unerwünschte Verhalten korrigiert werden.

- Testing - das Schreiben von automatisierten, codebasierten Tests. Diese stellen sicher, dass das Programm die definierten Anforderungen erfüllt. Gute Tests testen die verschiedenen funktionalen Komponenten sowohl unabhängig voneinander als auch ihrem Zusammenspiel und decken neben den Standard-Anwendungsfällen auch Ausnahme- und Randfälle ab. Die Definition von verständlichen und aussagekräftigen Testszenarien ist dabei von zentraler Bedeutung.

- Reviews und Pair-Programming - der team-interne Austausch über neuen Code. Vor dem Einsatz geschriebenen Codes in Produktivsystemen erfolgt als weiterer Teil der Qualitätssicherung eine Überprüfung mit einem:r Kolleg:in. Im Vier-Augen-Prinzip werden die Programmzeilen auf ihre technische und inhaltliche Korrektheit überprüft, Design- und Lösungsansätze werden diskutiert und bei Bedarf Verbesserungen und Veränderungen vorgenommen. Das stellt zusätzlich sicher, dass der Codingstil auch für andere Teammitglieder verständlich ist.

- Dokumentation - die Beschreibung des Codes und der Arbeitsergebnisse. Um das Wissen darüber zu bewahren und für andere zugänglich zu machen, wird während der Software-Entwicklung viel dokumentiert. Dies reicht von codenaher Dokumentation, bspw. erläuternden Kommentarzeilen im Code, über technische Diagramme, die das Zusammenspiel verschiedener Software-Komponenten visualisieren, bis hin zu codefernen Erklärungen, die algorithmische Ideen beschreiben (z.B. Pseudocode) oder technische Entscheidungen begründen.

Innerhalb der Software-Abteilung war ich Teil des **Team Optimization**, des Teams, das sich vorrangig mit den mathematischen Algorithmen beschäftigt, die logistische Optimierungsprobleme lösen sollen. Fokus meiner Arbeit lag also weder auf der Entwicklung von Benutzeroberflächen noch auf der Verwaltungen von großen Datenbanken, sondern auf der technischen Implementierungen von Heuristiken und Lösungsverfahren. Dadurch erweiterte sich die Liste der oben genannten Aufgaben noch um

- Analyse und mathematische Modellierung der Problemstellung - die Übersetzung von realen Fragestellungen in abstrakte Problembeschreibungen. Die vom Kunden formulierten Anforderungen, z.B. *„Bestimme für jeden Werksstandort (mindestens) einen Leergutstandort, und zwar so, dass die Transportkosten minimiert werden“*, werden auf ihren mathematischen Kern reduziert. So können viele Optimierungsfragen einer bereits gut untersuchten Problemfamilie zugeordnet werden, bspw. Kürzeste-Wege-Probleme, Zuweisungsprobleme, Bin-Packing-Probleme, etc. Desweiteren müssen kundenspezifische Anforderungen, bspw. *„Bestimmte Werke dürfen von höchstens zwei Leergutstandorten beliefert werden“*, modelliert werden. Dafür werden wohlbekannten mathematischen Formulierungen adaptiert und erweitert.

- Konzeption - die Entwicklung und Auswahl des Lösungsverfahrens. Für das gewählte Modell werden basierend auf bestehenden Heuristiken (wenn vorhanden) oder verbreiteten Verfahren (Greedy-Ansätze, Enumerationen, o.ä.) Lösungsansätze entwickelt und gegeneinander verglichen. Die gewählte Methode muss dabei darauf untersucht werden, ob sie alle nötigen Restriktionen zulässig und effizient behandeln kann,

- Performance-Analyse - die Untersuchung der entwickelten Lösung auf Güte bzgl. Lösungsqualität, Laufzeit und Speicherverbrauch. Dafür werden kundenähnliche und/oder in der Literatur vorhandene Input-Datensätze mit dem entwickelten Programm durchgerechnet, deren technische Performance evaluiert und die berechneten Ergebnisse mit Kunden-/Literatur-Erwartungen verglichen.

Im Folgenden möchte ich diese Aufgabenbereiche an zwei Beispielprojekten konkretisieren, an denen ich mitgewirkt habe.

## 3.2 Hauptversorgeroptimierung

Unter dem Stichwort der Hauptversorgeroptimierung beschäftigten wir uns im Team Optimization mit der Anforderung eines großen Automobilherstellers, dessen Leergutnetzwerk zu optimieren. Kern des Optimierungsproblem ist es, für jeden Produktionsstandort aus einer Menge von Lieferantenstandorten eine Teilmenge auszuwählen, die den Leergutbedarf des Produktionsstandorts abdecken sollen. Dabei müssen selbstverständliche viele Restriktionen beachtet werden, die die Zulässigkeit einer Lösung beeinflussen. Dazu zählen:

(A) Der Leergutbedarf eines Werkes muss vollständig abgedeckt sein.

(B) Jeder Leergutlieferant hat ein begrenztes Angebot an Leergut.

(C) Nicht jedes Werk kann von jedem Lieferanten beliefert werden.

Wir begannen die Bearbeitung damit, eine formale Problembeschreibung aufzustellen:

Gegeben:

$$\begin{aligned}
&\mathcal{P} \text{ eine Menge von Produktionsstandorten}\\
d(P) \in \mathbb{R}^+ \text{ für jedes } P \in \mathcal{P} &\text{ der Bedarf an Leergut eines Werkes}\\
&\mathcal{L} \text{ eine Menge von Leergutlieferanten}\\
s(L) \in \mathbb{R}^+ \text{ für jedes } L \in \mathcal{L} &\text{ das Angebot an Leergut eines Lieferanten}\\
\mathcal{V} \subseteq \mathcal{P} \times \mathcal{L} &\text{ eine Menge von unzulässigen Werk-Lieferant-Paaren}
\end{aligned}$$

Gesucht:

$$f\colon \mathcal{P} \times \mathcal{L} \to \mathbb{R}^+ \text{ eine Zuordnung von Quantitäten zu Werken-Lieferanten-Paaren}$$

sodass:

$$\begin{aligned}
\sum_{L \in \mathcal{L}} f(P, L) = d(P) &\text{ für alle } P \in \mathcal{P} \text{ (siehe Bedingung (A))}\\
\sum_{P \in \mathcal{P}} f(P, L) \leq s(L) &\text{ für alle } L \in \mathcal{L} \text{ (siehe Bedingung (B))}\\
f(P, L) = 0 &\text{ für alle } (P, L) \in \mathcal{V} \text{ (siehe Bedingung (C))}
\end{aligned}$$

Diese Formulierung des zugrundeliegenden mathematischen Problems offenbarte schnell die erste Herausforderung dieses Projektes: Die Existenz einer zulässigen Lösung ist nicht immer garantiert. Man betrachte Beispielsweise den Fall, dass der Gesamtbedarf aller Werke das Gesamtangebot aller Lieferanten übersteigt, d.h. $\sum_{P \in \mathcal{P}} d(P) \geq \sum_{L \in \mathcal{L}} s(L)$. Aber auch für weniger offensichtliche Fälle ist es nicht einfach zu entscheiden, ob eine zulässige Lösung existiert. Diese formale Hürde erwies sich in der konkreten Anwendung als weniger einschränkend, da wir von unserem Kunden fordern können, dass das aktuell Ist-Netzwerk den Beschränkungen (A) - (C) genügt und somit mindestens eine Lösung des Problems existiert.
Da wir so nun die Lösbarkeit des Problems sicherstellen konnten, fuhren wir fort, indem wir die Optimierungsziele des Kunden analysierten. Darin offenbarte sich die größte Komplexität der Anforderung: Der Kunde konnte uns keine klare Optimierungsobjektive benennen. Stattdessen erhielten wir den Auftrag, die folgenden Ziele alle gleichzeitig zu beachten:

(D) Die Gesamtkosten der Transporte sollen möglichst klein sein.

(E) Jedes Werk soll von möglichst wenigen Lieferanten beliefert werden.

(F) Möglichst wenige Werke sollen von mehr als einem Lieferanten beliefert werden. (genannt: Split am Werk)

(G) (Die Unterschiede zwischen dem aktuellen Ist-Zustand und der optimierten Soll-Lösung soll mäglichst klein sein.)

Wir formalisierten das nun multi-kriterielle Optimierungsproblem:

$$
\begin{aligned}
\text{Gegeben:} \quad & \mathcal{P} \text{ eine Menge von Produktionsstandorten} \\
& d(P) \in \mathbb{R}^+ \text{ für jedes } P \in \mathcal{P} \text{ der Bedarf an Leergut eines Werkes} \\
& \mathcal{L} \text{ eine Menge von Leergutlieferanten} \\
& s(L) \in \mathbb{R}^+ \text{ für jedes } L \in \mathcal{L} \text{ das Angebot an Leergut eines Lieferanten} \\
& \mathcal{V} \subseteq \mathcal{P} \times \mathcal{L} \text{ eine Menge von unzulässigen Werk-Lieferant-Paaren} \\
& c : \mathcal{P} \times \mathcal{L} \times \mathbb{R}^+ \to \mathbb{R}^+ \text{eine Kostenfunktion}
\end{aligned}
$$

$$
\begin{aligned}
\text{Finde} \quad & \min \sum_{P \in \mathcal{P}, L \in \mathcal{L}} c(P, L, f(P, L)) \text{ Kostenminimierung (D)} \\
& \min |\{(P, L) \in \mathcal{P} \times \mathcal{L} \mid f(P, L) > 0\}| \text{ Minimierung der Zuweisungen (E)} \\
& \min |\{P \in \mathcal{P} \mid |\{L \in \mathcal{L} \mid f(P, L) > 0\}| > 0\}| \text{ Minimierung der sogenannten Splits (F)} \\
& \text{sodass: } \sum_{L \in \mathcal{L}} f(P, L) = d(P) \text{ für alle } P \in \mathcal{P} \text{ (siehe Bedingung (A))} \\
& \sum_{P \in \mathcal{P}} f(P, L) \leq s(L) \text{ für alle } L \in \mathcal{L} \text{ (siehe Bedingung (B))} \\
& f(P, L) = 0 \text{ für alle } (P, L) \in \mathcal{V} \text{ (siehe Bedingung (C))} \\
& f(P, L) \in \mathbb{R}^+ \text{ für alle } (P, L) \in \mathcal{P} \times \mathcal{L} \text{ die Liefermenge von } L \in \mathcal{L} \text{ nach } P \in \mathcal{P}
\end{aligned}
$$

Es lässt sich schnell feststellen, dass sich diese verschiedenen Optimierungsziele widersprechen können, d.h. beispielsweise kann eine Lösung, die kostenoptimal ist, eine Lösung sein, die bezüglich der anderen Kriterien sehr schlecht ist. Üblicherweise verhindert man solche Konflikte indem,

- die verschiedenen Kriterien ineinander überführt werden. Z.B. wird jeder Split (siehe Bedingung (F)) mit Strafkosten belegt. Ein Lösungsalgorithmus, der Kosten minimieren möchte, wird so inzentiviert, auch Splits zu minimieren um Strafkosten zu vermeiden.
- die verschiedenen Kriterien gegeneinander priorisiert werden. Damit werden Kriterien mit geringer Priorität zu optionalen Nebenbedingungen.

Beide diese Ansätze waren für den Kunden allerdings nicht akzeptabel oder umsetzbar, sodass wir eine andere Strategie verfolgen mussten. Wir entschieden uns, das Problem Schritt für Schritt anzugehen.

In einer ersten Phase beschäftigten wir uns unabhängig voneinander damit, das zu Grunde liegende Problem besser zu verstehen. Dafür konzentrierten wir uns jeweils auf eines der Optimierungsziele und versuchten, in der einschlägigen Literatur ähnlich gelagerte Optimierungsprobleme zu finden und mit unserem Anwendungsfall zu vergleichen. Die verschiedenen Optimierungskriterien führten uns zu sehr verschiedenen Problembeschreibungen.

- Die Kostenminimierung liefert ein Problem aus der Familie der Transportprobleme
- Der Fokus auf die Minimierung der Zuweisungen liefert ein dem CWLP[5] ähnliches Problem
- Die Betrachtung der Split-Minimierung liefert ein GAP (General Assignment Problem)

Nach der individuellen Recherche trugen wir unsere Ergebnisse zusammen und stellten fest:

1. Bereits die einzelnen Subprobleme für **ein** Optimierungsziel sind zum Teil schwer (NP-schwer) zu lösen.
2. Für die untersuchten Subprobleme gibt es jeweils heuristische Lösungsverfahren.
3. In diese Heuristiken lassen sich die jeweils anderen Objektiven nicht als Nebenbedingungen einpflegen.

In Anbetracht der Unvereinbarkeit der Optimierungsziele entschieden wir uns, in einem iterativen Verfahren mit der Multikriteriellität umzugehen. Dafür entwickelten wir für jedes der Optimierungsziele zwei Heuristiken

- Eine Startheuristik, die ohne Betrachtung der anderen Kritieren eine möglichst gute (gemessen am gewählten Zielkriterium) Lösung finden soll.
- Eine Verbesserungsheuristik, die auf einer gegeben Startlösung kleine Änderungen vornimmt, um sie bzgl. des gewählten Zielkriteriums zu verbessern, ohne dabei die Lösung bzgl. anderer Zielfunktionen zu verschlechtern.

[5] Capacitated Warehouse Location Problem

So entwickelte ich bspw. verschiedene Greedy-Lösungsverfahren als Startheuristiken und ein Market-Negotiation-Verfahren als Verbesserungsheuristik. Diese Heuristiken konnten wir dann miteinander kombinieren, indem wir sie entweder nacheinander ausführen oder als Grundlage für Metaheuristiken (z.B. genetischen Verfahren[6]) nutzen.
Die große Anzahl an entwickelten Algorithmen und die noch größere Anzahl an Kombinationsmöglichkeiten derer ermöglichte es uns dann, dem Kunden zu jedem Optimierungslauf eine Vielzahl von pareto-optimalen Lösungen anzubieten.

Die so entstandenen Algorithmen wurden im Anschluss vielseitig getestet. Dazu zählen codenahe Tests der verschiedenen Implementierungen, Testläufe auf selbstgenerierten Datensätzen sowie Tests durch den Kunden auf seinen eigenen Realdaten. Auch Lasttests auf Massendaten wurden durchgeführt, um die Stabilität der Software in Bezug auf Speicherverbrauch und Laufzeit zu gewährleisten. In Ergebnis dieser Tests akzeptierte der Kunde unsere Software und sie ist dort seitdem im regelmäßigen, produktiven Einsatz.

## 3.3 Truckloadoptimierung

Ein weiteres Projekt, an dessen Umsetzung ich maßgeblich beteiligt war, ist die sogenannte Truckload-Optimierung. Grundlage des Optimierungspotentials ist ein Konflikt zwischen zwei Phasen in der Planung der Supply-Chain:

- Die Material-Planung. Ziel ist es, (a) Materialien stets rechtzeitig verfügbar zu haben (b) Materialien möglichst wenig lange in Lagern vorzuhalten und (c)
- Die Transport-Planung. Ziel ist es, Materialien möglichst selten und gebündelt zu verschicken, um beim Transport möglichst geringe Kosten zu verursachen.

In Abwägung dieser beiden Objektive soll die Truckload-Optimierung nun eine für die Materialplanung zulässige Lösung zu finden, die die Transportkosten berücksichtigt. Die daraus resultierende Fragestellung lässt sich folgendermaßen zusammenfassen: „Wie können die aus der Materialplanung entstandenen Transporte so um wenige Tage verschoben werden, dass die Transportkosten verringert werden?“

Erste Aufgabe war wieder eine formale Problembeschreibung und eine Analyse der erweiterten Restriktionen an die Lösung. Da in diesem Fall - anders als in vielen anderen logistischen Transportfragestellungen - keinen Mengen- oder Frequenzentscheidungen getroffen werden müssen, handelt es sich hier um ein diskretes Problem. Dies ermöglichte uns die Formulierung des Optimierungsproblems sowohl als eine graphentheoretische Aufgabe, als auch als Mixed Integer Programm (MIP).
Die dadurch gewonnenen Erkenntnisse nutzen wir dann in der folgenden Phase, in der wir verschiedene prototypische Lösungsverfahren entwickelten.

- Eine First-Fit Greedy Heuristik
- Eine auf dem Set-Covering Problem basierende Heuristik
- Ein Ruin-Create-Ansatz, der bestehende Lösungen verändern/verbessern kann
- Ein genetischer Algorithmus

Alle diese Ansätze waren noch nicht detailliert ausgearbeitet, sondern waren Grundlage für den nächsten Arbeitsschritt. Denn im Folgenden verglichen wir die entwickelten Ansätze gegeneinander. Dafür evaluierten wir Lösungsgüte und Laufzeit der verschiedenen Ansätze, getestet auf einer Menge von zum Teil Kundendaten um zum Teil zufällig generierten Testdaten. Als Bewertungsgrundlage/Maßstab für die Ergebnisse dienten dabei sowohl die Ist-Lösung (i.e. die Transporte, wie sie aus der Materialplanung hervorgehen) als auch Lösungen, die ein kommerzieller MIP-Solver für unsere MIP-Formulierung erzeugt. So konnten wir unsere Heuristiken nicht nur relativ zur Ist-Situation, sondern auch absolut zu einer optimalen Lösungen, bewerten (Ergebniss siehe Tabelle 1).

Nach Auswahl des Lösungsverfahrens begann der Prozess der technischen Implementierung. Dabei habe ich den prototypischen Code des Algorithmus nochmal großen Änderungen unterzogen, um

- weitere Kundenrestriktionen als Nebenbedingungen in den Algorithmus einzubinden
- allen Qualitätsstandards (Lesbarkeit, Modularität, Designprinzipien, etc.) an den Code zu genügen
- den Code vollständig unter automatisierte (sog. Unit-) Tests zu stellen
- das Programm den spezifischen Kundenanforderungen anzupassen
- den Code in die bestehende Code-Basis einzubinden

Die so entstanden Lösung ist bis heute bei verschiedenen Kunden in regelmäßigem Einsatz. Das modulare Design des Algorithmus ermöglichte es, die Software seitdem mehrfach um weitere Kundenanforderungen zu erweitern.

[6] Iteriertes Lösungsverfahren, bei dem mehrfach (a) eine Population von sehr vielen potentiellen Lösungen generiert wird (b) die besten Individuen einer Population ausgewählt, ggf. leicht verändert und (c) zur Generierung der nächsten Generarion genutzt werden.

| **Ansatz** | **Laufzeit** | **Lösungsqualität** | **Im Einsatz?** |
|---|---|---|---|
| First Fit Greedy | sehr schnell $\mathcal{O}(n \log n)$ | Kostenverbesserung, offensichtliche Abweichungen zum Optimum | ja |
| Set-Covering Heuristik | zu langsam $\mathcal{O}(n^2 \log n)$ | Leicht besser als Greedy | nein (zu langsam) |
| Ruin-Create Ansatz | schnell | behebt offensichtliche Abweichungen vom Optimum | ja (als Verbesserungs-heuristik) |
| Genetischer Algorithmus | zu langsam | kleine Verbesserungen ggü. Greedy + RuinCreate | nein (zu langsam) |

Tabelle 1: stark vereinfachte Zusammenfassung der Analyse der verschiedenen Ansätze

# 4 Fazit und Ausblick

Ich blicke mit großer Freude auf meine bisherige - und hoffentlich auch noch lange weitere - Zeit bei 4flow. Die Vielzahl als Projekten, an denen ich (zum Teil auch federführend) mitarbeiten darf, und die Vielseitigkeit der Aufgaben, ermöglichte es mir, mich in mehrerlei Hinsicht weiterzuentwickeln. Ich konnte nicht nur meine technischen Programmierfähigkeiten ausbauen, sondern habe auch sehr viel über den gesamten Prozess der Softwareentwicklung gelernt. Wie in den vorangegangenen Kapiteln beschrieben war es mir möglich, ein Softwareprodukt über seine gesamte Entstehungszeit zu begleiten - von der Kundenanforderung, über die Konzeption und Ideenfindung, die Implementierung bis hin zum Testen, der Dokumentation, der nachträglichen Anpassung und zur Auslieferung. Dies gibt mir einen ganzheitlichen Blick auf Softwareentwicklung.

Besonders glücklich bin ich darüber, dass ich durch den starken Bezug zu Optimierungsfragestellungen viele Anwendungen meines im Studium erlernten Wissens kennengelernt habe. Es ist mir möglich, meine Kompetenzen aus den Bereichen der Graphentheorie und der Algorithmischen Mathematik zur Lösung von realen, alltäglichen Problemen zu nutzen und so von der Theorie in die Praxis zu übersetzen.

In einer Kultur der großen gegenseitigen Wertschätzung, einer Philosophie der offenen Türen und in einem Team voller freundlicher Kolleg:innen ist es mir täglich eine Freude zu arbeiten.

# Bibliographie

# References


[4fl23] 4flow. *4flow Unternehmensfakten*. 2023. URL: `https://www.4flow.de/unternehmen/fakten.html`.

[Ber22] TU Berlin, ed. *StuPO B.Sc. Technomathematik*. 2022. URL: `https://www.static.tu.berlin/fileadmin/www/10000000/Studiengaenge/StuPOs/Fakultaet_II/Technomathe_B.Sc_2014.pdf`.