JOHANNES KEPLER
UNIVERSITÄT LINZ

Eingereicht von
Sebastian Schmalzer
k11803987

Angefertigt am
Research Institute for Symbolic Computation (RISC)

Betreuer und Beurteiler
Assoc. Univ.-Prof. DI Dr. Wolfgang Windsteiger

Zweitbetreuer
DI Dr. Michael Bögl

November 2021

VERFAHREN ZUM LÖSEN DES INVENTORY ROUTING PROBLEMS

Bachelorarbeit

zur Erlangung des akademischen Grades

Bachelor of Science

im Bachelorstudium

Technische Mathematik

JOHANNES KEPLER UNIVERSITÄT LINZ
Altenbergerstraße 69
4040 Linz, Österreich
www.jku.at
DVR 0093696

JOHANNES KEPLER
UNIVERSITÄT LINZ

Eidesstattliche Erklärung

Ich erkläre an Eides statt, dass ich die vorliegende Bachelorarbeit selbstständig und ohne fremde Hilfe verfasst, andere als die angegebenen Quellen und Hilfsmittel nicht benutzt bzw. die wörtlich oder sinngemäß entnommenen Stellen als solche kenntlich gemacht habe.

Linz, November 2021

Sebastian Schmalzer

JOHANNES KEPLER
UNIVERSITÄT LINZ

Zusammenfassung

Das Inventory Routing Problem stellt ein Optimierungsproblem in der Logistik dar, es verbindet dabei Bereiche der Lagerbestandsführung sowie der Tourenplanung. In der vorliegenden Arbeit werden neben der Einführung in die Problematik auch verschiedene Varianten und Charakterisierungen erläutert. In weiterer Folge wird das konkrete Distributionsproblem betrachtet, in dem ein Lieferant ein einzelnes Produkt an mehrere Kunden ausliefern muss, wobei deren Lager im Falle einer Anlieferung stets komplett aufgefüllt werden müssen. Die Lagerbestandsführung der Abnehmer liegt also im Verantwortlichkeitsbereich des Lieferanten. Das Ziel ist es, einen Lieferplan über einen vorgegebenen Zeitraum zu finden, sodass die Summe der Transport- und Lagerkosten beim Lieferanten sowie den Kunden minimal ist. Diese Problemstellung kann als ganzzahliges lineares Optimierungsproblem modelliert werden.

Im Zuge dieser Arbeit werden zwei verschiedene Lösungsmöglichkeiten für den Spezialfall betrachtet, dass ein einheitliches Produkt mit nur einem verfügbaren Fahrzeug ausgeliefert wird. Diese beiden Ansätze wurden in C# implementiert und anhand von Benchmarkproblemdaten aus der Literatur getestet. Beim ersten Ansatz wird versucht, eine exakte Lösung zu finden, indem das Optimierungsproblem einem Solver übergeben wird, welcher mit einem Branch-and-Bound Algorithmus arbeitet. Dieses Verfahren bringt jedoch den Nachteil mit sich, dass für Probleme mit größerer Kundenanzahl oder längerem Betrachtungszeitraum aufgrund der hohen Rechenkapazität keine Lösung gefunden werden kann. Der Gedanke beim zweiten Lösungsansatz ist, die optimale Lösung mithilfe einer Metaheuristik näherungsweise zu erreichen. Genauer gesagt wird eine Tabusuche behandelt, mit der auch für größere Problemstellungen eine Lösung gefunden werden kann. Anhand von Benchmark-Instanzen wird die Performance dieses Näherungsverfahrens getestet und geprüft, wo seine Stärken und Schwächen liegen.

JOHANNES KEPLER
UNIVERSITÄT LINZ

Inhaltsverzeichnis

1. Einleitung

Größere Unternehmen haben in den letzten Jahrzehnten bemerkt, dass eine globale optimierte Supply Chain zu einer beträchtlichen Reduktion von Kosten führen kann. Der Fortschritt in der Informations- bzw. Kommunikationstechnologie stärkt diesen Ansatz und führt dazu, dass die Kunden-Lieferanten-Beziehung immer engmaschiger miteinander verdrahtet wird. Einen der wohl wesentlichsten Berührungspunkte stellt dabei die Verknüpfung der Lagerbestandsführung und Disposition des Kunden mit der Belieferungsstrategie des Lieferanten dar. Die bessere Handhabung dieses Bindegliedes begründet die Strategie des *Vendor Managed Inventory*. Im Gegensatz zum herkömmlichen *Retailer Managed Inventory* zeigt sich hier der Lieferant für die Überwachung und Wiederbefüllung der Lagerbestände seiner Abnehmer verantwortlich. Dadurch sieht sich dieser mit der Problemstellung konfrontiert, einen Lieferplan zu finden, der sicherstellt, dass alle seine Kunden zeitgerecht beliefert werden, mit dem Ziel, die Summe der Transport- und Lagerkosten beider Parteien möglichst gering zu halten. Diese Aufgabenstellung ist besser bekannt als das *Inventory Routing Problem* und wird seit den 1980er Jahren untersucht. Einerseits kann dieses Problem als ein (gemischt) ganzzahliges lineares Optimierungsproblem modelliert werden und daher mit *MIP* (*Mixed Integer Programing*) exakt gelöst werden. Andererseits wird vor allem daran geforscht, geeignete Heuristiken zum Lösen des Inventory Routing Problems zu finden, um auch für sehr komplexe Problemstellungen günstige Lieferstrategien zu erhalten. Im Zuge dieser Bachelorarbeit werden sowohl ein exaktes als auch ein näherungsweises Lösungsverfahren in Form einer Tabusuche behandelt und gegenübergestellt.

Die Arbeit ist wie folgt aufgebaut: Kapitel 2 beinhaltet eine Einführung in die Thematik des Inventory Routing und es werden verschiedene Formen und Varianten näher erläutert. In Kapitel 3 wird die konkrete Problemstellung beschrieben und modelliert. Außerdem werden zwei Lösungskonzepte präsentiert, welche auf Branch-and-Bound bzw. Tabusuche basieren. In Kapitel 4 folgt eine Darstellung und Interpretation der Rechenergebnisse dieser beiden Lösungsansätze. Schlussendlich wird in Kapitel 5 noch ein Fazit gezogen bzw. ein Ausblick über die gewonnenen Erkenntnisse gegeben.

Motivationsbeispiel

Einer der ersten Artikel, in denen eine Form des Inventory Routing Problems behandelt wurde, stellt [6] aus dem Jahr 1983 dar, wo der Vertrieb von industriellen Gasen betrachtet wurde. Ein dort erstmals angeführtes Beispiel wird in [10] genauer behandelt und soll auch hier erste, einfache Grundprinzipien des Inventory Routing Problems veranschaulichen. Der Einfachheit halber spielen in diesem Beispiel die Lagerkosten noch keine Rolle, es werden lediglich die Transportkosten betrachtet. Es gilt zu entscheiden:

- Welcher Kunde wird zu welchem Zeitpunkt beliefert?
- Wie groß ist die jeweilige Liefermenge?
- Welche Route wird gewählt, um die Kunden zu erreichen?

Ziel dieses Beispiels ist es aufzuzeigen, wie die oben getroffenen Entscheidungen Einfluss auf die Transportkosten nehmen.

Angenommen die Zeit sei diskret in Tagen unterteilt und ein Lieferant soll sein Produkt an vier Kunden (gekennzeichnet durch die Indexmenge {1,2,3,4}) beliefern. Abbildung 1.1 zeigt die verfügbaren Verbindungen zwischen dem Lieferanten und den Kunden bzw. zwischen den Kunden untereinander, sowie die zugehörigen Fahrtkosten in Euro. Die Anzahl der zur Verfügung stehenden Lieferfahrzeuge sei unbegrenzt, jedoch besitzen alle Fahrzeuge eine Kapazitätsgrenze von $C = 5000$ Stück. Des Weiteren sind folgende Daten der Kunden bekannt:

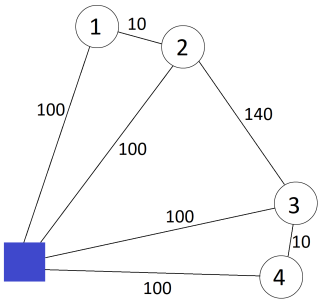


Abbildung 1.1.: Distributionsnetzwerk

Kunde	Verbrauch (Stück/Tag)	Lagerkapazität (Stück)
1	1000	5000
2	3000	3000
3	2000	2000
4	1500	4000

Das Ziel ist es nun, eine periodische Lieferstrategie mit minimalen Kosten zu finden, welche die Bedarfe der Kunden abdeckt und die Lager- bzw. Transportkapazitäten berücksichtigt.

Betrachtet man die Kundendaten etwas genauer, ist schnell erkennbar, dass bei Kunde 2 und Kunde 3 der tägliche Verbrauch gleich der maximalen Lagerkapazität ist. Demzufolge müssen diese beiden Kunden täglich beliefert werden.

Eine offensichtliche Lösung wurde in [10] gegeben, siehe Abbildung 1.2: Es werden täglich zwei Routen gefahren und somit zwei Lieferfahrzeuge genutzt. Das eine Lieferfahrzeug beliefert die nahe zueinander gelegenen Kunden 1 und 2 mit deren Tagesbedarfen, mit dem zweiten Fahrzeug wird selbiges für die Kunden 3 und 4 gemacht. Die Kosten der beiden Touren belaufen sich auf jeweils 210€. Somit verursacht diese Lösung Tageskosten von 420€.

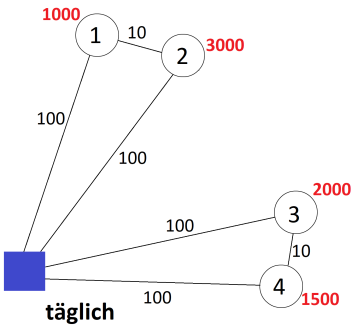


Abbildung 1.2.: Offensichtliche Lösung [10]

Schon in [10] wurde eine bessere Lösung gezeigt, indem man eine Periodizität von zwei Tagen betrachtet, siehe Abbildung 1.3. Am ersten Tag werden wieder zwei Touren gefahren, in denen Kunde 1 und 2 bzw. Kunde 3 und 4 zusammen beliefert werden. Jedoch überbringt man den Kunden 1 und 4 den doppelten Tagesverbrauch, was unter Berücksichtigung der Kapazitäten möglich ist. Dadurch müssen am nächsten Tag lediglich die Kunden 2 und 3 mit ihren jeweiligen Tagesbedarfen beliefert werden. Hierfür müssen zwar die hohen Transportkosten zwischen diesen beiden Kunden in Kauf genommen werden, im Gegenzug erspart man sich aber eine zweite Tour. Die Kosten dafür betragen 420€ am ersten Tag und 340€ am zweiten. Das ergibt durchschnittliche Tageskosten von 380€.

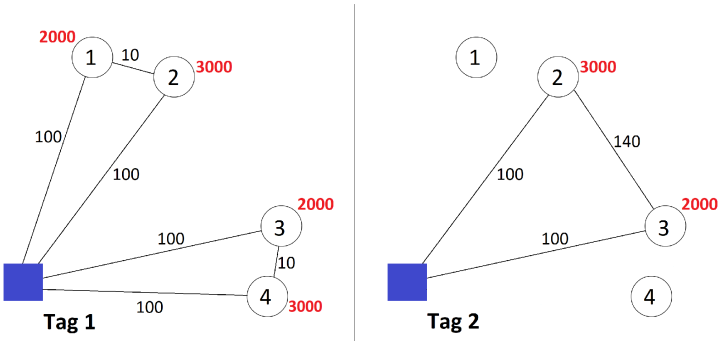


Abbildung 1.3.: Lösung mit einer Periodizität von zwei Tagen

Laut [10] ist auch bewiesen, dass das die bestmögliche Lösung für die gegebene Problemstellung ist. Das Beispiel verdeutlicht den Einfluss der Lieferzeitpunkte und -mengen auf die Transportkosten.

2. Das Konzept des Inventory Routing

In diesem Kapitel wird das Konzept des Inventory Routing allgemein beschrieben, dabei werden verschiedene Eigenschaften erläutert, die ein solches Konzept aufweisen kann.

2.1. Charakterisierung

Das Konzept des Inventory Routing ist durch jene Hauptcharakteristika gekennzeichnet, welche im Supply-Chain-Management als Strategie des *Vendor Managed Inventory (VMI)*, zu Deutsch „Lieferantengesteuererter Bestand", bekannt ist. Wie der Name vermuten lässt, besteht die Besonderheit dieser Distributionsstrategie darin, dass die Bestandsverwaltung und Nachschubdisposition bestimmter Artikel in die Verantwortlichkeit des Lieferanten übertragen wird. Voraussetzung dafür ist unter anderem die zeitgerechte Übermittlung der Bestands- und Bedarfszahlen des Kunden an den Lieferanten, der anhand dessen die Entscheidung über Liefermenge und Lieferzeitpunkt zu treffen hat. Aus Lieferantensicht wäre ein dauerhafter Echtzeit-Zugriff auf ebendiese Daten des Kunden noch besser [13].

Das VMI besitzt mehrere Vorteile gegenüber des traditionellen *Retailer Managed Inventory (RMI)*, zu deutsch „Kundengesteuerter Bestand". Wie der Name bereits vermuten lässt, obliegt bei Zweiterem die Entscheidung über Zeitpunkt und Menge einer Lieferung dem Kunden. Versucht der Lieferant im Konzept des RMI seine Auslieferungen zu optimieren, sind ihm gewissermaßen die Hände gebunden, da er zu sehr von den Entscheidungen seiner Kunden abhängig ist. Ein einfaches Beispiel zur besseren Veranschaulichung: Man stelle sich einen Lieferanten vor, welcher zwei sehr nahe zueinander angesiedelte Kunden zu beliefern hat. Angenommen beide bestellen wöchentlich eine kleinere Menge eines bestimmten Produkts und die Lagerkosten des jeweiligen Artikels seien vernachlässigbar. Während der eine Kunde die Lieferung für Dienstag anfordert, fragt der andere Kunde die Ware erst für Mittwoch an. Somit muss der Lieferant die selbe Tour zweimal pro Woche mit je einer halben Ladung zurücklegen. Würde man die Entscheidung bzgl. des Lieferzeitpunktes in den Bereich des Lieferanten übergeben (dieser steht dann in der

Verantwortung, dass die Lagerbestände seiner Kunden nie auslaufen dürfen), so würde dieser klarerweise eine gesammelte Lieferung organisieren, wodurch sich der Lieferservice für die beiden Kunden nicht ändern würde.

Beide Parteien können von solch einem Lieferkonzept profitieren: Auf der einen Seite genießt der Lieferant eine bessere Planungssicherheit, da für ihn die Bedarfe der Kunden besser vorhersehbar sind. Aus demselben Grund ist es ihm gegebenenfalls möglich, bei gleichbleibendem Servicegrad auch die eigenen Lagerbestände zu reduzieren. Eine Minderung der Transportkosten kann eine Folge einer besseren Ausnutzung der Fahrzeugkapazitäten sein, welche beispielsweise aus dem Zusammenlegen mehrerer Lieferungen resultieren kann. Durch Reduzierung der Lagerbestände bzw. Transportkosten ist es dem Lieferanten in weiterer Folge auch möglich, seinen Kunden einen besseren Preis für dessen Service anzubieten, womit schlussendlich beide Parteien Kosteneinsparungen erzielen würden. Auf Seiten des Kunden können Ressourcen im Logistikbereich eingespart werden und durch die Auslagerung der Lagerüberwachung und Disposition der jeweiligen Produkte zum Lieferanten wird weniger Personal benötigt. Des Weiteren profitiert die Kunden von erhöhter Produktverfügbarkeit: Er hat die Garantie des Lieferanten, stets genug Lagerbestand zur Verfügung zu haben [9].

Das Inventory Routing Problem (IRP) verbindet zwei Teilgebiete der Logistik: einerseits den Bereich der Touren- und Routenplanung, andererseits den Bereich des Lagerbestandsmanagements. Das Ziel eines IRPs besteht folglich darin, jene Lieferstrategie zu finden, welche die gesamten Transportkosten oder die gesamten Lagerkosten (sowohl jene des Lieferanten als auch die des Kunden) oder die Summe der beiden minimiert. Letzteres wird zumeist der Fall sein. Eine solche Lieferstrategie stellt einen Plan dar, der folgende Fragen beantworten muss [10]:

- Welcher Kunde wird zu welchem Zeitpunkt beliefert?
- Wie groß sind die Mengen, die die jeweiligen Kunden zum jeweiligen Zeitpunkt erhalten?
- Welche Routen werden gewählt, um die zu beliefernden Kunden mit den verfügbaren Fahrzeugen abzudecken?

2.2. Varianten des Inventory Routing

Im folgenden Abschnitt wird in Anlehnung an [10] und [2] auf verschiedene Varianten des IRP eingegangen. Es werden unterschiedliche Eigenschaften vorgestellt, welche ein IRP aufweisen kann.

Struktur eines Netzwerks

Es wird ein Distributionsnetzwerk behandelt, welches aus einem Lieferanten und einem oder mehreren Kunden besteht. Der Lieferant stellt gewissermaßen den Ursprung des Verteilungsflusses dar. Da die Möglichkeit besteht, dass der Lieferant von mehreren Standorten aus beliefert (z. B. falls er mehrere Lagerdepots besitzt), unterscheidet man die Struktur eines Netzwerkes je nach Quantität der Niederlassungen des Lieferanten bzw. Anzahl der Kunden zwischen [2] :

- *one-to-one*
- *one-to-many*
- *many-to-one*
- *many-to-many*

Auslieferungszeit

Die möglichen Auslieferungszeiten eines IRP sind:

- *Stetig*: Eine Auslieferung kann jederzeit ab einem Startzeitpunkt vorgenommen werden
- *Stetig mit Mindestzeitdifferenz*: Eine Auslieferung kann jederzeit ab einem Startzeitpunkt vorgenommen werden, jedoch darf die Zeitdifferenz zwischen zwei aufeinanderfolgenden Auslieferungen eine vorgegebene Mindestzeit nicht unterschreiten.
- *Diskret*: Auslieferungen können nur an Vielfachen einer Mindestzeitdifferenz stattfinden. Da diese Mindestzeitdifferenz ohne Beschränkung der Allgemeinheit auf eins normiert werden kann, werden Lieferungen nur zu diskreten Zeitpunkten vorgenommen [10].

Planungshorizont

Beim Planungshorizont wird zwischen zwei Arten unterschieden:

- *Unendlich*: In diesem Fall zielt das IRP darauf ab, einen Langzeitplan zur erstellen, um so Entscheidungen über Fuhrpark, Anzahl der Fahrer usw. besser treffen zu können.
- *Endlich*: Die Länge des Planungshorizont hängt von dem individuellen Problem ab, welches gelöst werden soll [10].

Strategie der Lagerauffüllung

Die Liefer- und Lagerstrategie nimmt Einfluss auf die Menge und auf den Zeitpunkt der Lieferung und somit in weiterer Folge auch auf die zu fahrenden Routen der Lieferfahrzeuge. Beispiele für solche Strategien sind:

- *Zero-Inventory-Ordering*: Ein Kunde wird genau dann beliefert, wenn dessen Lagerstand des jeweiligen Produktes den Nullpunkt erreicht.
- *Periodic*: Es muss eine Periode P gefunden werden und jede Aktion, die zum Zeitpunkt t (wobei $0 \leq t < P$) stattfindet, wird zu den Zeiten $t + kP$ (für $k = 1, 2, \dots$) wiederholt.
- *Full-Load*: Lieferungen werden nur dann vollzogen, wenn die Fahrzeuge vollbeladen werden können.
- *Direct-Shipping*: Es werden ausschließlich direkte Lieferungen vom Lieferanten zum jeweiligen Kunden vorgenommen. Somit sind all jene Routen unzulässig, welche Lieferungen zu mehr als einem Kunden vorsehen.
- *Maximum-Level*: Jeder Kunde hat eine maximale Lagerkapazität für das zu liefernde Produkt definiert. Wird ein Kunde mit diesem Produkt beliefert, muss die gelieferte Menge dementsprechend gewählt werden, dass die maximale Lagerkapazität nicht überschritten wird.
- *Order-up-to-Level*: Diese Vorgehensweise stellt einen Spezialfall der Maximum Level Strategie dar. Bei jeder vorgenommenen Lieferung eines Produktes wird die Liefermenge genau so gewählt, dass die maximale Lagerkapazität erreicht wird.
- *Fixed-Partition*: Die Menge der Kunden wird in mehrere disjunkte Teilmengen geteilt. Jede Teilmenge wird dann separat und unabhängig von den anderen beliefert.

Da eine Lieferroute somit nur Kunden beinhaltet, die in der selben Teilmenge liegen, findet die Partitionierung typischerweise nach geographischen Kriterien statt.

- *Partition-based*: Diese Strategie stellt eine Verallgemeinerung der Fixed Partition dar, denn auch hier werden die Kunden in mehrere Teilmengen geteilt. Eine Route darf wieder Kunden von ausschließlich einer Teilmenge beinhalten, zusätzlich aber auch Kunden einer zuvor festgelegten Kombination zweier oder mehrerer Teilmengen [10].

Größe bzw. Zusammensetzung des Fuhrparks

Bei der Größe des Fuhrparks unterteilt man je nach Anzahl v der verfügbaren Fahrzeuge [2]:

- *single-vehicle*: $v = 1$
- *multi-vehicle*: $v > 1$
  - *homogen*: Einheitliche Fahrzeuge (gleiche Kapazitäten)
  - *heterogen*: Unterschiedliche Fahrzeuge (verschiedene Kapazitäten)
- *unconstrained*: $v \in \mathbb{N}$
  - *homogen*: Einheitliche Fahrzeuge (gleiche Kapazitäten)
  - *heterogen*: Unterschiedliche Fahrzeuge (verschiedene Kapazitäten)

Größe der Produktpalette

Ähnlich wie beim Fuhrpark, kann man ein IRP auch je nach Größe p der Produktpalette unterscheiden [2]:

- *Single-Product Problem*: $p = 1$
- *Multi-Product Problem*: $p > 1$

Zielvorgabe

Die optimale Lösung eines IRP hängt von der gewählten Zielvorgabe ab, diese kann sein [10]:

- *Minimierung der Transportkosten*: Falls der Entscheidungsträger nur für die Transportkosten verantwortlich ist oder falls die Lagerkosten im Vergleich zu den Transportkosten vernachlässigbar sind
- *Minimierung der Lagerkosten*: Falls der Fokus auf dem Lagermanagement liegt, in diesem Fall sind häufige Anlieferungen zu erwarten
- *Minimierung der Summe von Transport- und Lagerkosten*: Falls der Entscheidungsträger für die Gesamtkosten verantwortlich ist

Entscheidungsarten

Bei reinen Tourenplanungsproblemen sind sowohl die anzuliefernden Kunden als auch die zu liefernden Mengen vorgegeben. Somit beziehen sich die noch zu treffenden Entscheidungen allesamt auf die Routen der Fahrzeuge. Solche Entscheidungen werden daher als *decisions over space* bezeichnet. Da bei IRPs stets die Zeiten und Mengen der Lieferungen im Entscheidungsraum inkludiert sein müssen, können die folgenden beiden Entscheidungsarten klassifiziert werden [2]:

- *Decisions over time and quantity*: Hier sind die Routen bereits festgelegt, es wird lediglich über den Lieferzeitpunkt bzw. der Liefermenge entschieden. Diese Entscheidungsart ist beispielsweise bei der *Direct-Shipping-Strategie* implizit gegeben: Da pro Lieferung nur ein Kunde abgedeckt werden darf, ist die Route mit dem direkten Weg vom Lieferanten zum Kunden und wieder zurück bereits fixiert.
- *Decisions over time, quantity and space*: In diesem Fall müssen Entscheidungen über die Zeitpunkte der Lieferungen zu den einzelnen Kunden sowie den zugehörigen Liefermengen getroffen werden. Gleichzeitig müssen aber auch noch die Touren, welche die Lieferfahrzeuge zum jeweiligen Zeitpunkt zurücklegen sollen, festgelegt werden.

3. Lösungsansätze für das Inventory Routing Problem

In der weiteren Arbeit beschränken wir uns auf den folgenden Spezialfall eines IRP:

- Single-Product
- Single-Vehicle
- Order-up-to-Level Strategie

Dieser Spezialfall des IRP kann in der Praxis durchaus in Erscheinung treten, denkt man zum Beispiel an die Treibstoffindustrie beim Beliefern von Tankstellen.

Die Problemstellung und Formulierung des mathematischen Modells sind in Anlehnung an [9] erarbeitet.

3.1. Problembeschreibung

Wie in der Mathematik üblich, bezeichnet im folgenden Kapitel A^n die Menge aller Tupel der Länge $n \in \mathbb{N}_0$ und $A^{m \times n}$ die Menge aller $m \times n$-Matrizen über eine Menge A. Sowohl bei einem Tupel $a \in A^n$ als auch bei einer Matrix $b \in A^{m \times n}$ gilt (falls nicht explizit anders erwähnt), dass die Indizierung bei 0 beginnt. Der Zugriff auf das erste Element erfolgt also über den Index 0, damit können $a \in A^n$ und $b \in A^{m \times n}$ wie folgt schematisch dargestellt werden:

$$a = (a_0, a_1, \ldots, a_{n-1}) \qquad b = \begin{pmatrix} b_{0,0} & b_{0,1} & \cdots & b_{0,n-1} \\ b_{1,0} & b_{1,1} & \cdots & b_{1,n-1} \\ \vdots & \vdots & \ddots & \vdots \\ b_{m-1,0} & b_{m-1,1} & \cdots & b_{m-1,n-1} \end{pmatrix}$$

Außerdem bezeichnen wir mit $A^{\leq n} := \bigcup_{i=0}^{n} A^i$ die Menge aller Tupel mit maximaler Länge n über A. Mit $|.|$ kennzeichnen wir die Länge eines Tupel, somit gilt:

- $|a| = n$, falls $a \in A^n$ und
- $|a| \leq n$, falls $a \in A^{\leq n}$.

In der folgenden Problemstellung wird davon ausgegangen, dass es sich beim jeweiligen Produkt um ein Stückgut handelt, daher werden Verbräuche, Kapazitäten, etc. stets ganzzahlig betrachtet, in der Realität muss das natürlich nicht so sein. Bei der Problemstellung haben wir folgende Größen gegeben:

- Anzahl der Kunden $n \in \mathbb{N}$, sodass die einzelnen Kunden von 1 bis n durchnummeriert werden können und $\mathcal{K} := \{1, 2, \ldots, n\}$ die Menge aller Kunden bezeichnet. Weiters wird mit 0 der Lieferant und mit $\mathcal{M} := \{0, 1, 2, \ldots, n\}$ die Menge aller im Distributionsnetzwerk vorkommenden Knoten bezeichnet.
- Zeithorizont $H \in \mathbb{N}$, sodass die Kunden über einen diskreten endlichen Zeitraum $\mathcal{T} := \{1, 2, \ldots, H\}$ beliefert werden müssen. Zu jedem Zeitpunkt $t \in \mathcal{T}$ kann jeder Kunde maximal einmal beliefert werden.
- Fahrzeugkapazität $C \in \mathbb{N}$.
- Anfangslagerbestände $A \in \mathbb{N}^{n+1}$ sodass A_s den Lagerbestand beim Knoten $s \in \mathcal{M}$ zum Zeitpunkt 0 entspricht.
- Produktions- und Verbrauchsraten $r \in \mathbb{N}^{n+1}$, sodass
  - r_0 der Produktionsmenge pro Zeitperiode beim Lieferanten entspricht und
  - r_s der Verbrauchsmenge pro Zeitperiode beim Kunden $s \in \mathcal{K}$ entspricht.

  In dieser Arbeit werden die Produktions- und Verbrauchsraten als konstant angenommen, was in der Realität nicht der Fall sein muss.
- Lagerkosten pro Einheit $h \in \mathbb{R}^{n+1}$, sodass für alle $s \in \mathcal{M} : h_s > 0$ und
  - h_0 den Lagerkosten pro Einheit beim Lieferanten und
  - h_s den Lagerkosten pro Einheit beim Kunden $s \in \mathcal{K}$ entspricht.

- Maximale Lagerkapazität $u \in \mathbb{N}^n$, wobei hier $u = (u_1, \ldots, u_n)$ mit dem Index 1 beginnt, sodass u_s die maximale Menge kennzeichnet, welche der Kunde $s \in \mathcal{K}$ einlagern kann. Beim Lieferanten wird die Lagerkapazität als unbeschränkt angenommen.

- Transportkosten $c \in \mathbb{R}^{(n+1)\times(n+1)}$, sodass $c_{i,j}$ die Transportkosten bezeichnet, welche beim Befahren der Kante zwischen den beiden Knoten $i, j \in \mathcal{M}$ anfallen. Es gilt $c_{i,i} = 0$ für alle $i \in \mathcal{M}$ und $c_{i,j} = c_{j,i} > 0$ für alle $i, j \in \mathcal{M}$ mit $i \neq j$. In dieser Arbeit werden die Transportkosten als symmetrisch und unabhängig von der Liefermenge angenommen, was in der Realität nicht der Fall sein muss. Somit kann c geschrieben werden als

$$c = \begin{pmatrix} 0 & c_{0,1} & \cdots & c_{0,n} \\ c_{1,0} & 0 & \cdots & c_{1,n} \\ \vdots & \vdots & \ddots & \vdots \\ c_{n,0} & c_{n,1} & \cdots & 0 \end{pmatrix} = \begin{pmatrix} 0 & c_{0,1} & \cdots & c_{0,n} \\ c_{1,0} & 0 & \cdots & c_{1,n} \\ \vdots & \vdots & \ddots & \vdots \\ c_{n,0} & c_{n,1} & \cdots & 0 \end{pmatrix}^T.$$

Folgende Parameter sind in der Problemstellung gesucht:

- Liefermengen $x \in \mathbb{N}^{n\times H}$, wobei hier die Zeilen- und Spaltenindizes mit 1 beginnen, sodass $x_{s,t}$ die Liefermenge zum Kunden $s \in \mathcal{K}$ zum Zeitpunkt $t \in \mathcal{T}$ kennzeichnet, also $x = \begin{pmatrix} x_{1,1} & \cdots & x_{1,H} \\ \vdots & \ddots & \vdots \\ x_{n,1} & \cdots & x_{n,H} \end{pmatrix}$.

- Lagerbestände $I \in \mathbb{N}^{(n+1)\times(H+1)}$, sodass $I_{s,t}$ dem Lagerbestand beim Knoten $s \in \mathcal{M}$ zum Zeitpunkt $t \in \mathcal{T} \cup \{H+1\}$ entspricht. Dabei starten die Spaltenindizes mit 1, also $I = \begin{pmatrix} I_{0,1} & I_{0,2} & \cdots & I_{0,H+1} \\ I_{1,1} & I_{1,2} & \cdots & I_{1,H+1} \\ \vdots & \vdots & \ddots & \vdots \\ I_{n,1} & I_{n,2} & \cdots & I_{n,H+1} \end{pmatrix}$.

- Routen $R \in (\mathcal{K}^{\leq n})^H$, wobei hier $R = (R_1, \ldots, R_H)$ mit Index 1 beginnt. Eine Route beginnt und endet immer beim Lieferanten, dieser wird in der Routendarstellung nicht explizit angegeben. Die Route $R_t = (R_{t,1}, R_{t,2}, \ldots, R_{t,|R_t|}) \in \mathcal{K}^{\leq n}$ mit $R_{t,1}, R_{t,2}, \ldots, R_{t,|R_t|} \in \mathcal{K}$ startet wiederum mit Index 1 und entspricht der Tour vom Lieferanten zu $R_{t,1}$, weiter zu $R_{t,2}$ usw. bis zu $R_{t,|R_t|}$ und zurück zum Lieferanten.

Die in der Problemstellung gesuchten Größen unterliegen folgenden Nebenbedingungen:

- Die Liefermengen x müssen so gewählt werden, dass
  1. zu jedem Zeitpunkt $t \in \mathcal{T}$ der Lagerbestand beim Lieferanten mindestens so groß wie die Summe der zu diesem Zeitpunkt zu den Kunden ausgelieferten Mengen ist,
  2. die Lagerbestände der Kunden nie negativ sind,
  3. der Lagerbestand eines Kunden komplett aufgefüllt wird, falls eine Lieferung zu diesem stattfindet und
  4. die Gesamtliefermenge zu jedem Zeitpunkt $t \in \mathcal{T}$ die Fahrzeugkapazität nicht überschreitet.
- Die Lagerbestände I müssen so gewählt werden, dass
  5. sich der Lagerbestand beim Lieferanten zum künftigen Zeitpunkt $t+1$ für $t \in \mathcal{T}$ aus der Summe des aktuellen Lagerbestandes zum Zeitpunkt t und der Differenz zwischen aktuell produzierter Menge und ausgelieferter Gesamtmenge ergibt und
  6. sich der Lagerbestand bei einem Kunden $s \in \mathcal{K}$ zum künftigen Zeitpunkt $t+1$ für $t \in \mathcal{T}$ aus der Summe des aktuellen Lagerbestandes von s zum Zeitpunkt t und der Differenz zwischen der zum Zeitpunkt t angelieferten bzw. verbrauchten Menge ergibt.
- Die Routen R müssen so gewählt werden, dass
  7. ein Kunde $s \in \mathcal{K}$ in R_t für $t \in \mathcal{T}$ genau dann genau einmal vorkommt, falls s zum Zeitpunkt t beliefert wird, d.h. $x_{s,t} > 0$.

Diese Problemstellung mit ebendiesen sieben Nebenbedingungen wird in weiterer Folge mit IRP bezeichnet. Da in der Regel mehrere mögliche Lösungen für IRP existieren ist es sinnvoll, das zugehörige Optimierungsproblem zu betrachten, in dem man auf der Suche nach jener möglichen Lösung ist, welche die geringsten Gesamtkosten (Summe der Lagerkosten beim Lieferanten und den Kunden sowie Transportkosten) verursacht.

3.2. Beispiel: Inputdaten inklusive möglicher Lösung

Zur Illustration einer expliziten Problemstellung, wird in diesem Abschnitt eine mögliche Lösung für eine Benchmark-Instanz, wie sie in [9] und [8] behandelt werden, präsentiert. Es gelten die Bezeichnungen aus Kapitel 3.1. Die gegebenen Inputdaten sehen wie folgt aus:

- Kundenanzahl $n = 5$
- Zeithorizont $H = 3$
- Fahrzeugkapazität $C = 456$
- Lieferanten-/ Kundendaten:

		A	u	h	r
Lieferant	0	674	-	0.30	304
Kunde	1	87	174	0.48	87
Kunde	2	86	172	0.37	86
Kunde	3	65	130	0.40	65
Kunde	4	106	159	0.49	53
Kunde	5	26	39	0.33	13

- Kostenmatrix:

$$c = (c_{i,j})_{i,j=0,\dots,5} = \begin{pmatrix} 0 & 510.80 & 163.37 & 53.49 & 405.58 & 404.75 \\ 510.80 & 0 & 384.30 & 521.28 & 335.60 & 464.25 \\ 163.37 & 384.30 & 0 & 149.19 & 244.83 & 268.76 \\ 53.49 & 521.28 & 149.19 & 0 & 379.44 & 364.59 \\ 405.58 & 335.60 & 244.83 & 379.44 & 0 & 131.10 \\ 404.75 & 464.25 & 268.76 & 364.59 & 131.10 & 0 \end{pmatrix}$$

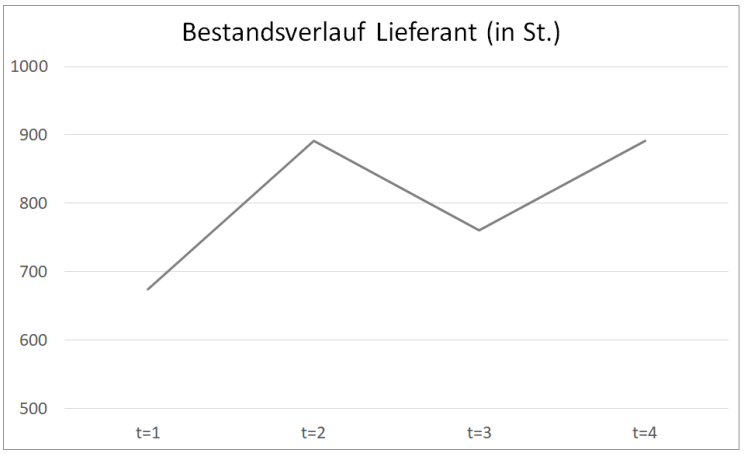
Eine intuitive und mögliche Lösung wäre, die folgenden Routen zu fahren:

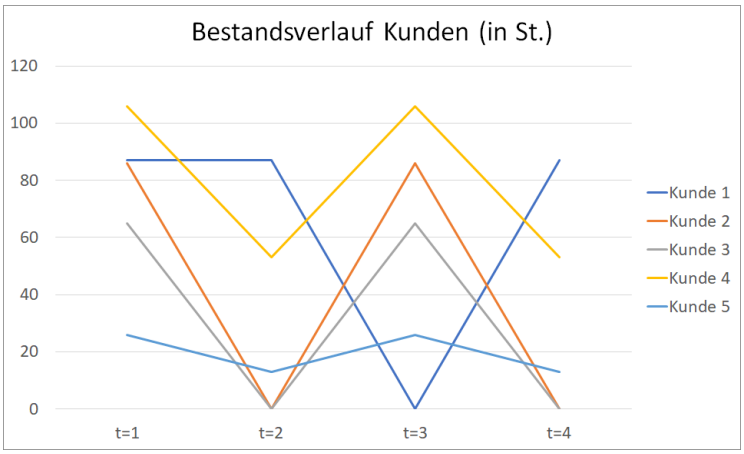
- $R_1 = (1)$
- $R_2 = (2, 4, 5, 3)$
- $R_3 = (1)$

Die zugehörigen Liefermengen belaufen sich auf $x = \begin{pmatrix} 87 & 0 & 174 \\ 0 & 172 & 0 \\ 0 & 130 & 0 \\ 0 & 106 & 0 \\ 0 & 26 & 0 \end{pmatrix}$.

Die Lagerbestände des Lieferanten bzw. der Kunden belaufen sich auf $I = \begin{pmatrix} 674 & 891 & 761 & 891 \\ 87 & 87 & 0 & 87 \\ 86 & 0 & 86 & 0 \\ 65 & 0 & 65 & 0 \\ 106 & 53 & 106 & 53 \\ 26 & 13 & 26 & 13 \end{pmatrix}$

und können grafisch wie folgt veranschaulicht werden:





Die Transportkosten für die drei gefahrenen Routen belaufen sich gesamt auf 3000.58 GE.

Die Lagerkosten werden über den Zeitraum $\mathcal{T} \cup \{H+1\}$ berechnet, wobei der Zeitpunkt $H+1$ inkludiert wird, um die Konsequenzen für die Entscheidungen, welche im letzten Zeitpunkt H getroffen wurden, im Ergebnis miteinzubeziehen. Somit belaufen sich die Lagerkosten auf 965.10 GE beim Lieferanten und 422.48 GE bei den Kunden. Die Gesamtkosten dieser zulässigen Lösung betragen daher 4388.16 GE.

3.3. Modellierung des Optimierungsproblems

Mit Hilfe der Hilfsvariablen

- $z \in (\{0,1\}^{(n+1)})^H$ mit
$$z_{i,t} = \begin{cases} 1 & \dots \text{falls } i \in \mathcal{M} \text{ zur Zeit } t \in \mathcal{T} \text{ besucht wird} \\ 0 & \dots \text{sonst} \end{cases}$$
- $y \in ((\{0,1,2\}^{\leq n})^{\leq n})^H$ mit
$$y_{i,j}^t = \text{Anzahl der Fahrten zwischen den beiden Knoten } i,j \in \mathcal{M} \text{ mit } j < i \text{ zum Zeitpunkt } t \in \mathcal{T}\,.$$

kann das in Kapitel 3.1 beschriebene IRP kann als ganzzahliges lineares Optimierungsproblem für die Variablen x, y und z formuliert werden. Die Hilfsvariablen y und z dienen der Darstellung der gesuchten Routen mit Hilfe von ganzzahligen Variablen. Die Einträge in z sorgen dafür, dass aus y^t die Route R_t rekonstruiert werden kann.

Das nachfolgende Modell basiert auf jenem, das in [9] beschrieben ist. Die Reihenfolge und Nummerierung der einzelnen Formeln entspricht jener, wie sie auch in der Implementierung vorgenommen wurde und dient dem besseren Zusammenhang zwischen Modellierung und Programmcode:

Nebenbedingungen

Zur korrekten und vollständigen Modellierung der Problemstellung werden folgende Nebenbedingungen benötigt:

1. *Übergangsbedingung des Lagerbestandes beim Lieferanten*
   (bezieht sich auf Nebenbedingung 5. aus Abschnitt 3.1)

$$\begin{aligned} I_{0,1} &= A_0 \\ I_{0,t+1} &= I_{0,t} + r_0 - \sum_{s \in \mathcal{K}} x_{s,t} \qquad \forall t \in \mathcal{T} \end{aligned} \tag{3.1}$$

2. *Bedingung gegen Auslauf des Lagerbestandes beim Lieferanten*
   (bezieht sich auf Nebenbedingung 1. aus Abschnitt 3.1)

$$I_{0,t} \geq \sum_{s \in \mathcal{K}} x_{s,t} \qquad \forall t \in \mathcal{T} \tag{3.2}$$

3. *Übergangsbedingung der Lagerbestände bei den Kunden*
   (bezieht sich auf Nebenbedingung 6. aus Abschnitt 3.1)

$$\begin{aligned} I_{s,1} &= A_s && \forall s \in \mathcal{K} \\ I_{s,t+1} &= I_{s,t} + x_{s,t} - r_s && \forall s \in \mathcal{K} \quad \forall t \in \mathcal{T} \end{aligned} \tag{3.3}$$

4. *Bedingung gegen Auslauf der Lagerbestände bei den Kunden:*
   (bezieht sich auf Nebenbedingung 2. aus Abschnitt 3.1)

$$I_{s,t} \geq 0 \qquad \forall s \in \mathcal{K} \quad \forall t \in \mathcal{T} \cup \{H+1\} \tag{3.4}$$

5. *Order-up-to-Level Bedingungen*
   (beziehen sich auf Nebenbedingung 3. aus Abschnitt 3.1)

$$x_{s,t} \geq u_s z_{s,t} - I_{s,t} \qquad \forall s \in \mathcal{K} \quad \forall t \in \mathcal{T} \tag{3.5}$$

$$x_{s,t} \leq u_s - I_{s,t} \qquad \forall s \in \mathcal{K} \quad \forall t \in \mathcal{T} \tag{3.6}$$

$$x_{s,t} \leq u_s z_{s,t} \qquad \forall s \in \mathcal{K} \quad \forall t \in \mathcal{T} \tag{3.7}$$

6. *Bedingungen für die Routenplanung*
   Zur Garantie, dass zu jedem Zeitpunkt $t \in \mathcal{T}$ eine zulässige Route gefunden wird, die alle Kunden beinhaltet, welche zu demjenigen Zeitpunkt beliefert werden sollen:

   a) *Bedingung zur Inkludierung des Lieferanten und Beachtung der Fahrzeugkapazität*
      (bezieht sich auf Nebenbedingung 4. und 7. aus Abschnitt 3.1)

$$\sum_{s \in \mathcal{K}} x_{s,t} \leq C z_{0,t} \qquad \forall t \in \mathcal{T} \tag{3.8}$$

b) *Bedingung zur Sicherstellung einer Rundreise*
Falls ein Knoten $i \in \mathcal{M}$ in der Route zum Zeitpunkt $t \in \mathcal{T}$ besucht wird, so müssen exakt zwei Kanten gefahren werden, welche i inkludieren.

$$\sum_{\substack{j \in \mathcal{M} \\ j<i}} y_{i,j}^t + \sum_{\substack{j \in \mathcal{M} \\ j>i}} y_{j,i}^t = 2z_{i,t} \qquad \forall i \in \mathcal{M} \quad \forall t \in \mathcal{T} \tag{3.9}$$

c) *Subtour-Eliminationsbedingung*
Diese Bedingung verhindert, dass innerhalb einer Route eine geschlossene Rundreise auf einer Teilmenge der Kunden vorkommt.

$$\sum_{i \in \mathcal{S}} \sum_{\substack{j \in \mathcal{S} \\ j<i}} y_{i,j}^t \leq \sum_{i \in \mathcal{S}} z_{i,t} - z_{k,t} \qquad \forall \mathcal{S} \subseteq \mathcal{K} \quad \forall t \in \mathcal{T} \quad \forall k \in \mathcal{S} \tag{3.10}$$

Die Idee dieser Subtour-Eliminationsformulierung ist, dass in eine Route die n Knoten beinhaltet genau dann keine Rundreise stattfindet, wenn die Anzahl der Kanten $n-1$ ist. In Abbildung 3.1 ist eine Veranschaulichung für die Formulierung für den Fall von $n = 3$ gegeben. Laut Formel 3.10 wäre zusätzlich noch der Spezialfall zu berücksichtigen, dass es einen Kunden k gibt, der nicht in der Route enthalten ist. Ist die Bedingung jedoch bereits für einen Kunden erfüllt der in der Route besucht wird, so ist dieser Spezialfall automatisch erfüllt, da dann die rechte Seite der Ungleichung um 1 größer wäre.

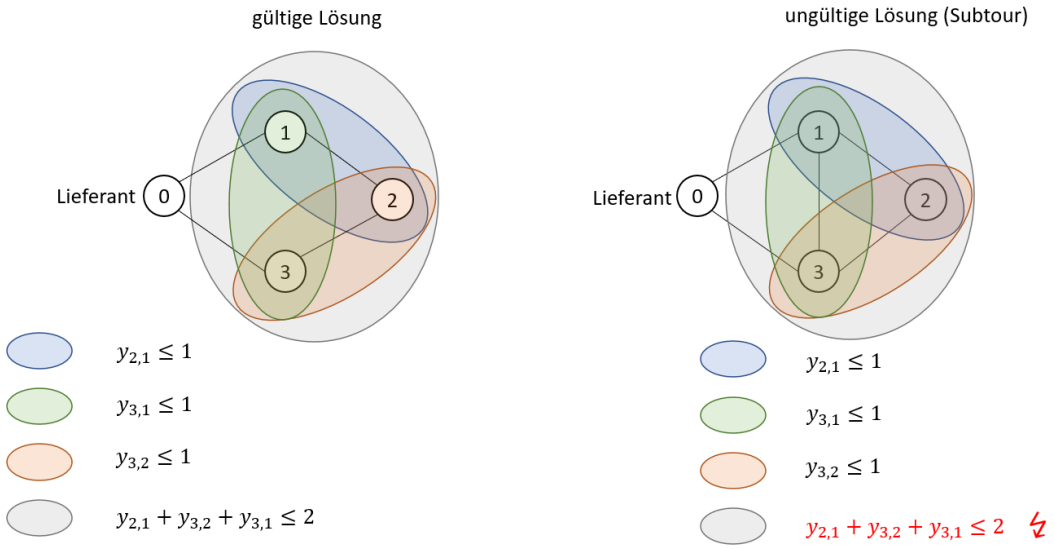


Abbildung 3.1.: Veranschaulichung der Subtour-Elimination

7. *Nichtnegativitäts- und Ganzzahligkeitsbedingungen*

$$y_{i,j}^t \in \{0,1\} \qquad \forall i \in \mathcal{K} \quad \forall j \in \mathcal{K}, j < i \quad \forall t \in \mathcal{T} \tag{3.11}$$

$$y_{i,0}^t \in \{0,1,2\} \qquad \forall i \in \mathcal{K} \quad \forall t \in \mathcal{T} \tag{3.12}$$

$$z_{i,t} \in \{0,1\} \qquad \forall i \in \mathcal{M} \quad \forall t \in \mathcal{T} \tag{3.13}$$

Zielfunktion

Ziel des Optimierungsproblems ist es, eine Lösung zu finden, sodass die Gesamtkosten minimal sind. Die Gesamtkosten setzen sich aus den folgenden Teilen zusammen:

- Lagerkosten beim Lieferanten L_0:

$$L_0(H,h,I) = \sum_{t=1}^{H+1} h_0 I_{0,t}$$

- Lagerkosten bei den Kunden L_K:

$$L_K(n,H,h,I) = \sum_{s=1}^{n} \sum_{t=1}^{H+1} h_s I_{s,t}$$

- Transportkosten T:

$$T(n,H,c,y) = \sum_{i=1}^{n} \sum_{j=0}^{i-1} \sum_{t=1}^{H} c_{i,j} y_{i,j}^t$$

Wie bereits in Abschnitt 3.2 erwähnt, wird bei den Lagerkosten der Zeitpunkt $H+1$ miteinbezogen, um die Konsequenzen der Entscheidungen, welche am letzten Zeitpunkt H getroffen wurden, im Ergebnis miteinzubeziehen.

Die Zielfunktion des ganzzahligen linearen Optimierungsproblems lautet somit:

$$f(n,H,c,h,I,y) = L_0(H,h,I) + L_K(n,H,h,I) + T(n,H,c,y) \quad \rightarrow min \tag{3.14}$$

Bei näherer Betrachtung des Optimierungsproblems ist schnell erkennbar, dass die Anzahl der im Modell auftretenden Variablen n_{var} stark von der betrachteten Anzahl an Kunden n und der Zeitspanne H abhängt. Berechnet werden kann diese mit

$$n_{var} = \frac{1}{2}n^2H + \frac{7}{2}nH + n + 2H + 1$$

Die Anzahl der Variablen erhöht sich also rasant bei Betrachtung eines längeren Zeithorizonts H und noch stärker bei steigender Kundenanzahl n. Es ist daher davon auszugehen, dass in Rechenstudien von Implementierungen zum Lösen des Inventory Routing Problems die Programmlaufzeit bei größerer Anzahl an zu betrachtenden Kunden oder Zeitperioden stark ansteigen wird.

3.4. Exaktes Lösen eines Mixed Integer Linear Program

Das in Abschnitt 3.3 beschriebene Modell ist ein sogenanntes *Mixed Integer Linear Program (MILP)*, zu Deutsch „gemischt-ganzzahliges lineares Optimierungsproblem“. Ein MILP ist dadurch gekennzeichnet, dass die Variablen (oder zumindest ein Teil davon) ganzzahlige Werte annehmen müssen. Des Weiteren gilt, dass sowohl die Zielfunktion als auch die gesamten Nebenbedingungen linear sind. Da in dem hier betrachteten Problem der Spezialfall vorliegt, dass ausschließlich ganzzahlige Variablen gesucht werden, kann von einem *ILP*, also *Integer Linear Program* gesprochen werden.

Zum Lösen solcher linearen Optimierungsprobleme gibt es verschiedene Algorithmen. Im Rahmen dieser Bachelorarbeit wurde das mathematische Modell des Optimierungsproblems in C# mithilfe des frei zugänglichen Softwarepakets *Google OR Tools* implementiert, welches zum Lösen von MILP einen *Branch-and-Bound* Algorithmus anwendet.

Vorgehensweise

Die dabei auftretenden Aufgabenstellungen umfassen unter anderem das Deklarieren des MILP Solvers, Einlesen der Problemdaten sowie das Einbinden der gesamten Nebenbedingungen und der Zielfunktion.

Nach vollständiger Implementierung des Optimierungsmodells, wendet der Solver im Hintergrund einen Branch-and-Bound Algorithmus an, um die optimalen Werte für die

einzelnen Variablen zu bestimmen. Der Branch-and-Bound Algorithmus stellt eine kombinatorische Prozedur zum Lösen von ganzzahligen bzw. gemischt-ganzzahligen linearen Optimierungsproblemen dar. Wie der Name bereits sagt, basiert der Algorithmus auf dem Aufteilen und Begrenzen des Lösungsraumes. Wie in Abbildung 3.2 dargestellt, entsteht durch das fortwährende Aufspalten des Originalproblems (gekennzeichnet durch den obersten Knoten) in vereinfachte Subprobleme ein Wurzelbaum (*Branch-and-Bound-Baum*) [3].

Die Vorgehensweise in jedem einzelnen Knoten des Baumes sieht wie folgt aus: Zunächst wird eine Relaxation des Optimierungsproblems gelöst (beispielsweise mit dem Simplexverfahren), in der die Ganzzahligkeitsbedingung der gesuchten Variablen außer Acht gelassen werden. Dabei können folgende drei Fälle auftreten:

1. Die Lösung ist ganzzahlig, dann haben wir eine beste zulässige Lösung gefunden
2. Die Lösung ist unzulässig, dann kann der Knoten verworfen werden
3. Die Lösung besitzt nicht-ganzzahlige Variablen, obwohl diese ganzzahlig sein sollten

Im letzteren Fall wird eine nicht-ganzzahlige Variable gewählt und zwei neue Subprobleme gebildet (*Branching*), indem man der Variable einmal den nächstgrößeren und einmal den nächstkleineren ganzzahligen Wert zuweist. Welche nicht-ganzzahlige Variable dafür gewählt wird, wird durch eine zuvor festgelegte Verzweigungsregel bestimmt. Ist die aus der Relaxation gewonnene Schranke schlechter, sprich in unserem Fall größer, als der Wert der besten bisher bekannten Lösung, so kann der jeweilige Knoten verworfen werden *(Bounding)*. In Abbildung 3.2 beschreiben die blau schraffierten Knoten diesen Fall. Eine optimale Lösung ist erreicht, wenn keine offenen Knoten mehr im Baum vorhanden sind [7].

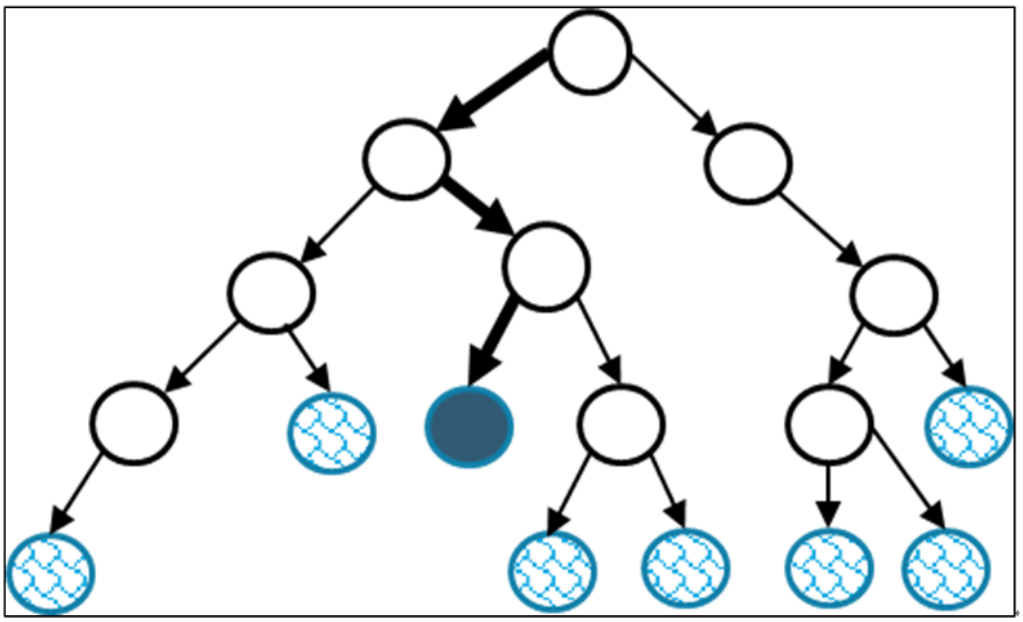


Abbildung 3.2.: Veranschaulichung des Branch-and-Bound Algorithmus [12]

Anwendung der ILP Implementierung

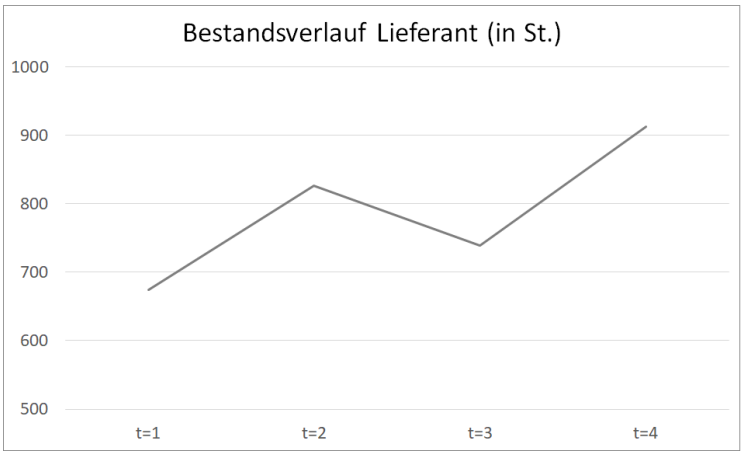
Mithilfe des implementierten ILP Modells kann der Solver nun die optimale Lösung des in Abschnitt 3.2 betrachteten Problems ermitteln. Betrachten wir exakt die gleichen Inputdaten, so gibt das Programm eine Lösung zurück, in der die folgenden Routen gefahren werden:

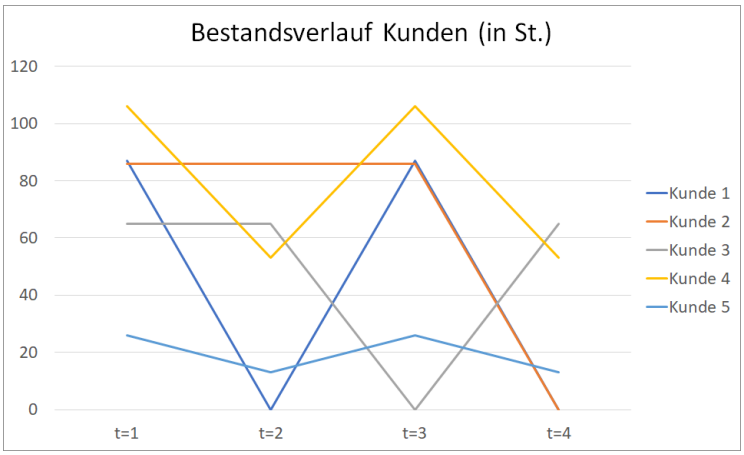
- $R_1 = (2, 3)$
- $R_2 = (1, 4, 5, 2)$
- $R_3 = (3)$

Die zugehörigen Liefermengen belaufen sich auf $x = \begin{pmatrix} 0 & 174 & 0 \\ 86 & 86 & 0 \\ 65 & 0 & 130 \\ 0 & 106 & 0 \\ 0 & 26 & 0 \end{pmatrix}$.

Die Lagerbestände des Lieferanten bzw. der Kunden entwickeln sich dabei wie folgt:

$$I = \begin{pmatrix} 674 & 827 & 739 & 913 \\ 87 & 0 & 87 & 0 \\ 86 & 86 & 86 & 0 \\ 65 & 65 & 0 & 65 \\ 106 & 53 & 106 & 53 \\ 26 & 13 & 26 & 13 \end{pmatrix}$$





Die Transportkosten für die drei Routen in der optimalen Lösung belaufen sich auf 1882.66 GE. Des Weiteren fallen Lagerkosten von 945.9 GE beim Lieferanten und 438.54 GE bei den Kunden an. Damit betragen die Gesamtkosten dieser durch den Branch-and-Bound Algorithmus gefundenen, optimalen Lösung 3267.10 GE. Verglichen mit den Kosten von 4388.16 GE der intuitiven Lösung aus Abschnitt 3.2 bedeutet das eine Einsparung von 25.55%.

3.5. Näherungsweises Lösen durch Metaheuristik

Dem Vorteil des Erlangen einer exakten Lösung durch den Branch-and-Bound Algorithmus steht die Befürchtung gegenüber, dass ebendiese nur für Probleme mit einer geringen Größenordnung gefunden werden kann. Wie aus der Formel zur Berechnung der Variablenanzahl n_{var} in Kapitel 3.3 hervorgeht, vergrößert sich diese rasant mit steigender Kundenanzahl n bzw. bei Betrachtung eines längeren Zeithorizont H. Da dies in der Realität zumeist auch der Fall sein wird, besteht großes Interesse darin, geeignete approximative Verfahren zum Lösen solcher Probleme zu ermitteln.

Die Tabusuche als Abwandlung der lokalen Suche

Die Tabusuche ist eine Metaheuristik und stellt eine Erweiterung der Idee der lokalen Suche dar. Lokale Suchverfahren sind durch das Grundprinzip gekennzeichnet, dass ausgehend von einer Startlösung versucht wird, eine bessere Lösung zu finden, indem eine lokale Änderung vorgenommen wird. Für jede Lösung L wird also eine Menge von Lösungen $N(L)$, genannt *Nachbarschaft*, durchsucht, die sich von L nur um eine lokale Änderung unterscheiden. Formal gesprochen wird ein Suchraum S durchlaufen, indem man sich Schritt für Schritt von einer Lösung $L \in S$ zur nächsten $L' \in N(L) \subseteq S$ bewegt. Weiters wird mit O die zugehörige Operation bezeichnet, um von L nach L' zu gelangen, also $L' = O(L)$. Ein wichtiges Merkmal der lokalen Suche stellt die Eigenschaft dar, dass in jedem Iterationsschritt nur eine bzgl. der Zielfunktion $f : S \to \mathbb{R}$ bessere Lösung als die Aktuelle zugelassen wird. Bei Minimierungsproblemen, wie es beispielsweise das Inventory Routing Problem ist, wo die Zielfunktionswerte Kosten darstellen, muss also $f(L') < f(L)$ gelten. Eine Verbesserung kann somit als Abstieg der Lösung bezeichnet werden [3].

Das grundlegende Prinzip kann anhand von Algorithmus 1 beschrieben werden.

Die lokale Suche bricht genau dann ab, wenn in der Nachbarschaft keine bessere Lösung mehr gefunden werden kann. Somit kann das zurückgegebene L^* als lokale Lösung bezeichnet werden, da alle weiteren Lösungen in dessen Nachbarschaft entweder gleich gut oder schlechter sind.

Die Tabusuche unterscheidet sich von der lokalen Suche dahingehend, dass, falls keine bessere Lösung in der betrachteten Nachbarschaft vorhanden ist, nicht abgebrochen, sondern auch eine schlechtere neue Lösung gestattet wird. Außerdem wird während der

Algorithmus 1 Lokale Suche

Eingabe: Suchraum S, Zielfunktion $f : S \to \mathbb{R}$
Ausgabe: Lokales Minimum $L^* \in S$ bzgl. f

1: Generiere eine Startlösung $L^* \in S$
2: Wiederhole:
Solange $\exists L \in N(L^*) : f(L) < f(L^*)$:
Sei $L' \in N(L^*)$ mit $f(L') < f(L^*)$
Setze $L^* \leftarrow L'$
3: Gib L^* zurück

Suche eine sogenannte *Tabuliste* mitgeführt, um eine bestimmte Anzahl von kürzlich erreichten Lösungen im Gedächtnis zu behalten. Anders ausgedrückt kommt man damit dem Interesse nach, Kenntnis darüber zu haben, auf welchem Pfad man zur aktuellen Lösung gelangt ist. Die Idee dahinter ist, die auf der Tabuliste beschriebenen Lösungen im unmittelbaren weiteren Verlauf zu verbieten und dadurch Zyklen in der Iterationsfolge zu verhindern [11].

Die Idee der Tabusuche wurde 1986 von Fred Glover erstmals veröffentlicht, der grundlegende Algorithmus kann wie folgt beschrieben werden:

Algorithmus 2 Tabusuche

Eingabe: Suchraum S, Zielfunktion f, Länge der Tabuliste l
Ausgabe: $L_{best} \in S$ mit $f(L_{best})$ möglichst klein

1: Generiere eine Startlösung $L^* \in S$
Setze $L_{best} \leftarrow L^*$
2: Erstelle Tabuliste $T = (L^*)$
3: Wiederhole, solange Abbruchskriterium nicht erfüllt:
4: Erstelle eine Nachbarschaft $N(L^*) \subseteq S$ um die aktuelle Lösung L^*, sodass für alle $L \in N(L^*)$ gilt $\neg Contains(T, L')$
5: Sei $L' \in N(L^*)$ mit $f(L') \leq f(L)$ für alle $L \in N(L^*)$
6: $Append(T, L')$
7: Falls $|T| > l$: lösche den ältesten Tabulisteneintrag, also $RemoveAt(T, 0)$
8: Setze $L^* \leftarrow L'$
9: Falls $f(L^*) < f(L_{best})$: setze $L_{best} \leftarrow L^*$
10: Gib L_{best} zurück

Tabusuche zum Lösen des Inventory Routing Problems

In diesem Kapitel wird die Art der Tabusuche näher erläutert, wie sie im Zuge dieser Bachelorarbeit implementiert wurde. Auch wenn bei der Durchführung einer Tabusuche die Grundidee stets dieselbe ist, gibt es verschiedenste Ausführungen und Abwandlungen sie zu implementieren [11]. Allesamt bringen Vor- und Nachteile mit sich. Es gilt einen Kompromiss zu finden, sodass die Tabusuche problemspezifisch bestmögliche Ergebnisse in annehmbarer Rechenzeit erzielt. Im Falle der nachfolgend beschriebenen Tabusuche wurden beispielsweise nicht gesamte Lösungen in der Tabuliste hinterlegt, sondern nur die getätigten Operationen, durch welche die kürzlich erhaltenen Lösungen erreicht wurden. Behält man sich lediglich die vollzogenen Operationen im Gedächtnis, so besteht zwar unter Umständen die Gefahr, dass ein Zyklus in der Iterationsfolge nur kurzfristig verhindert werden kann, jedoch wäre das Abspeichern von kompletten Lösungen aus Gründen der Rechenzeit und Speicherkapazität nur schwer umsetzbar. Die Funktionsweise der vorgenommenen Implementierung, welche zum Teil an der in [8] dargestellten Idee angelehnt ist, wird nun folgend beschrieben und anschließend zusammengefasst in Algorithmus 3 dargestellt.

Zuallererst ist der zu betrachtende Suchraum festzulegen, wobei zwei verschiedene Arten von Lösungen unterscheiden. Dafür wird mit IRP_z eine abgeschwächte Problemstellung von IRP eingeführt, indem bei IRP_z die Restriktionen 1 und 4 aus Abschnitt 3.1 vernachlässigt werden. Damit gestattet man im Problem IRP_z einen negativen Lagerbestand beim Lieferanten und die Überschreitung der Fahrzeugkapazität. Eine Lösung $L := (I, R, x)$ bezeichnet man als

- *zulässig*, falls L eine Lösung von IRP_z ist bzw.
- *realisierbar*, falls L darüber hinaus eine Lösung von IRP ist.

Als Suchraum S für die Tabusuche wird die Menge der weniger eingeschränkten zulässigen Lösungen gewählt. Da man nach Durchführung des Algorithmus logischerweise an einer realisierbaren Lösung interessiert ist, wird die Zielfunktion im Vergleich zur Formel 3.14 aus Abschnitt 3.3 um zwei Strafterme erweitert. Diese beiden Strafterme quantifizieren Verletzungen gegen den Auslauf des Lagerbestandes beim Lieferanten bzw. die Fahrzeugkapazität und werden wie folgt ermittelt:

- $V_1(H, I) = \sum_{t=1}^{H+1} max(-I_{0,t}, 0)$
- $V_2(n, H, x, C) = \sum_{t=1}^{H} max(\sum_{s=1}^{n} x_{s,t} - C, 0)$

Die Berechnung der Lagerkosten L_0 beim Lieferanten und L_K bei den Kunden erfolgt gleich wie in Abschnitt 3.3. Die Berechnung der Transportkosten erfolgt hingegen über die Routen $R = (R_1, \ldots, R_H) \in (\mathcal{K}^{\leq n})^H$ mit

$$T(H, R, c) = \sum_{r=1}^{H} \Big(\sum_{i=1}^{|R_r|-1} c_{R_{r,i}, R_{r,i+1}} + c_{0, R_{r,1}} + c_{R_{r,|R_r|}, 0} \Big).$$

Die Zielfunktion, die minimiert werden soll, lautet somit:

$$\begin{aligned} f_{Tabu}(n, H, h, c, C, I, R, x) = L_0(H, h, I) + L_K(n, H, h, I) + T(R, c) \\ + \alpha V_1(I, H) + \beta V_2(n, H, x, C) \end{aligned}$$

Dabei stellen α und β sogenannte Straffaktoren dar, welche Verletzungen im Hinblick auf die Realisierbarkeit einer Lösung in der Zielfunktion bestrafen sollen. Hinreichend große Faktoren α und β sorgen dafür, dass der Algorithmus vermeiden wird, Lösungen mit ebensolchen Verletzungen zurückzugeben. Der Einfachheit halber wurden diese Faktoren in der Implementierung als konstant angenommen.

Für die Generierung einer Startlösung $L^* = (I^*, R^*, x^*)$ werden die Kunden $1, \ldots, n$ der Reihe nach einzeln betrachtet und deren Zeitpunkte für Anlieferungen so spät wie möglich gewählt, sodass ein Auslauf des Lagerbestandes gerade noch verhindert werden kann. Damit ergeben sich die Liefermengen x^* und die Lagerbestände I^* aus der Order-up-to-Level Strategie. Die Routen R^* in der Startlösung werden so gewählt, dass an den jeweiligen Zeitpunkten die Kunden aufsteigend nach deren Nummer beliefert werden. Offensichtlich ist so eine Lösung immer zulässig, aber nicht zwingendermaßen realisierbar.

Nun wird eine Nachbarschaft um die aktuelle Lösung erstellt, was mithilfe der sogenannten *Nachbarschaftsoperatoren* geschieht. Im Zuge der hier vorgenommenen Implementierung wurden drei Operatoren entwickelt: *Remove*, *Insert* und *Move*. Deren Funktionsweise für die Order-up-to-Level Strategie kann wie folgt dargestellt werden:

- *Remove:* Wähle einen beliebigen Zeitpunkt $t \in \mathcal{T}$, an dem eine Auslieferung an zumindest einen Kunden stattfindet. Wähle einen beliebigen Index $k \in \{1, \ldots, |R_t|\}$ und entferne den Kunden $s := R_{t,k}$ aus der Route R_t. Falls der entfernte Kunde s zu einem späteren Zeitpunkt $t_1 \in \mathcal{T}$ mit $t_1 > t$ nochmals beliefert wird, wird die Liefermenge x_{s,t_1} um die zuvor entfernte Menge erhöht. Die Lagerbestände I werden

entsprechend aktualisiert. Das Entfernen einer Belieferung darf nur dann durchgeführt werden, wenn dadurch kein Auslauf des Lagerbestandes beim jeweiligen Kunden entsteht.

- *Insert:* Wähle einen beliebigen Zeitpunkt $t \in \mathcal{T}$ und einen beliebigen Kunden $s \in \mathcal{K}$, sodass zum Zeitpunkt t noch keine Lieferung zu s stattfindet, also $\neg Contains(R_t, s)$. Bestimme weiters einen beliebigen Index $k \in \{1, \ldots, |R_t|\}$ und füge s an der k-ten Stelle in der Route R_t ein. Die Liefermenge $x_{s,t}$ für diese neu eingefügte Anlieferung ergibt sich aus der Differenz zwischen der maximalen Lagerkapazität u_s und dem aktuellen Lagerbestand $I_{s,t}$. Falls Kunde s zu einem späteren Zeitpunkt $t_1 \in \mathcal{T}$ mit $t_1 > t$ nochmals beliefert wird, wird die dortige Liefermenge x_{s,t_1} um $x_{s,t}$ vermindert. Die Lagerbestände I werden entsprechend aktualisiert.

- *Move:* Wähle einen beliebigen Zeitpunkt $t \in \mathcal{T}$, an dem eine Auslieferung an zumindest einen Kunden stattfindet. Wähle einen beliebigen Index $k \in \{1, \ldots, |R_t|\}$ und entferne den Kunden $s := R_{t,k}$ aus der Route R_t. Falls der entfernte Kunde s zu einem späteren Zeitpunkt $t_1 \in \mathcal{T}$ mit $t_1 > t$ nochmals beliefert wird, wird die Liefermenge x_{s,t_1} um die zuvor entfernte Menge erhöht. Wähle einen beliebigen Zeitpunkt $t_{neu} \in \mathcal{T}$, an dem noch keine Lieferung zu s stattfindet, also $\neg Contains(R_{t_{neu}}, s)$ und einen beliebigen Index $k_{neu} \in \{1, \ldots, |R_{t_{neu}}|\}$ sodass die Lieferung nicht wieder an der exakt selben Stelle eingefügt wird, also $\neg AND(Equals(t, t_{neu}), Equals(k, k_{neu}))$. Füge s an der k_{neu}-ten Stelle in der Route R_t ein. Die Liefermenge $x_{s,t_{neu}}$ für diese neu eingefügte Anlieferung ergibt sich aus der Differenz zwischen der maximalen Lagerkapazität u_s und dem aktuellen Lagerbestand $I_{s,t_{neu}}$. Falls Kunde s zu einem späteren Zeitpunkt nochmals beliefert wird, wird die dortige Liefermenge um $x_{s,t_{neu}}$ vermindert. Die Lagerbestände I werden entsprechend aktualisiert. So eine Verschiebung einer Belieferung darf nur dann durchgeführt werden, wenn dadurch kein Auslauf des Lagerbestandes beim jeweiligen Kunden entsteht.

Eine festgelegte Größe m gibt an, wie viele benachbarte Lösungen jeder einzelne Operator erstellen soll (falls so viele zulässige Nachbarlösungen existieren). Die Nachbarschaft $N(L^*)$ ergibt sich dann aus der Vereinigung dieser Mengen. Dabei wird eine Lösung L nur dann in die Nachbarschaft aufgenommen, falls deren zugehörige Operation $O : \mathcal{S} \to \mathcal{S}$ mit $L = O(L^*)$ nicht in der Tabuliste vorhanden ist. Das Verbot der Durchführung einer in der Tabuliste enthaltenen Operation zielt darauf ab, ein Hin- und Herspringen zwischen Lösungen (zumindest kurzfristig) zu verhindern.

Im nächsten Schritt wird die neue Lösung $L' \in N(L^*)$ bestimmt, sodass L' kleiner als alle anderen in $N(L^*)$ gemessen an der die Zielfunktion f_{Tabu}. Diese Strategie zur Bestimmung des nächsten Iterationswertes wird in der Nachbarschaftssuche als *Best Improvement Strategy* bezeichnet. Ein anderes Konzept wäre die *First Improvement Strategy*, bei der die erste Lösung mit verbessertem Zielfunktionswert übernommen wird. Zwar weist die Best Improvement Strategy meist einen höheren Evaluierungsaufwand auf, da dafür die gesamte Nachbarschaft untersucht werden muss, jedoch sind die dadurch erlangten Abstiege dementsprechend größer zu erwarten [3].

Nachdem eine entsprechende neue Lösung L' gefunden wurde, wird die Inverse der für L' benötigten Operation O^{-1} in die Tabuliste aufgenommen. Überschreitet die Tabuliste ein vorgegebene konstante Länge l, so wird der älteste Eintrag entfernt. Somit besteht die Tabuliste stets aus den Umkehroperationen der letzten l Iterationsschritte. Der Algorithmus endet, sobald $MaxIter$ Iterationen durchgeführt wurden.

Algorithmus 3 Tabusuche für das IRP

Eingabe: Konstanten lt. Problembeschreibung aus Abschnitt 3.1, Suchraum S, Länge der Tabuliste l, maximale Iterationsanzahl $MaxIter$
Ausgabe: $L_{best} = (I_{best}, R_{best}, x_{best}) \in S$ mit $f_{Tabu}(n, H, h, c, C, I_{best}, R_{best}, x_{best})$ möglichst klein

1: Generiere eine Startlösung $L^* = (I^*, R^*, x^*) \in S$
Setze $L_{best} \leftarrow L^*, i = 0$
2: Erstelle Tabuliste $T = ()$
3: Wiederhole, solange $i < MaxIter$:
4: $i = i + 1$
5: Erstelle eine Nachbarschaft $N(L^*) \subseteq S$ um die aktuelle Lösung L^*, sodass für alle $L \in N(L^*)$ gilt, dass die zugehörige Operation O mit $L = O(L^*)$ nicht in der Tabuliste vorhanden ist, also $\neg Contains(T, O)$
6: Sei $L' = (I', R', x') \in N(L^*)$ mit $f(n, H, h, c, C, I', R', x') \leq f(n, H, h, c, C, I, R, x)$ für alle $L = (I, R, x) \in N(L^*)$
7: $Append(T, O^{-1})$
8: Falls $|T| > l$, lösche den ältesten Tabulisteneintrag, also $RemoveAt(T, 0)$
9: Setze $L^* \leftarrow L'$
10: Falls $f_{Tabu}(n, H, h, c, C, I^*, R^*, x^*) < f_{Tabu}(n, H, h, c, C, I_{best}, R_{best}, x_{best})$:
setze $L_{best} \leftarrow L^*$
11: Gib L_{best} zurück

4. Rechenergebnisse

Sowohl das ILP Modell mit dem entsprechenden Solver zum exakten Lösen als auch die Metaheuristik in Form der vorgestellten Tabusuche zum näherungsweisen Lösen wurden in C# implementiert und auf einem AMD Ryzen 2,1 GHz und 8 GB RAM Notebook ausgeführt.

Getestet wurden die beiden Programme an den Benchmarkinstanzen, welche in [1] frei zugänglich sind. Diese Instanzen wurden unter anderem auch für die Tests in [9] und [8] verwendet. Sie weisen folgende Charakteristika auf:

- der Zeithorizont entspricht entweder $H = 3$ oder $H = 6$;
- die Anzahl der Kunden entspricht $n = 5k$ für $k = 1, \ldots, 6$;
- die Produktmenge $r_{i,t}$ welche der Kunde i zum Zeitpunkt t benötigt, ist konstant über die Zeit t, das heißt $\forall t \in \mathcal{T}$ gilt $r_{i,t} = r_i$ und stellt eine Zufallszahl aus dem Intervall $[10, 100]$ dar;
- die Produktmenge $r_{0,t}$, welche beim Lieferant zum Zeitpunkt $t \in \mathcal{T}$ neu verfügbar wird, entspricht $\sum_{i \in \mathcal{K}} r_i$;
- die maximale Lagerkapazität u_i beim Kunden $i \in \mathcal{K}$ entspricht $r_i g_i$, wobei $g_i \in \{2, 3\}$ zufällig gewählt ist und die Anzahl der Zeitperioden angibt, in denen u_i verbraucht wird;
- der Anfangslagerbestand $I_{0,0}$ beim Lieferanten entspricht $\sum_{i \in \mathcal{K}} u_i$;
- der Anfangslagerbestand $I_{i,0}$ beim Kunden $i \in \mathcal{K}$ entspricht $u_i - r_i$;
- die Lagerkosten pro Einheit h_i beim Kunden $i \in \mathcal{K}$ sind zufällig aus dem Intervall $[0.1, 0.5]$ bzw. $[0.01, 0.05]$ gewählt;
- die Lagerkosten h_0 beim Kunden entsprechen 0.3 falls h_i aus $[0.1, 0.5]$ bzw. 0.03 falls h_i aus $[0.01, 0.05]$ erzeugt wurde.

- die Fahrzeugkapazität C entspricht $\frac{3}{2} \sum_{i \in \mathcal{K}} r_i$;
- die Transportkosten $c_{i,j}$ zwischen den Knoten $i, j \in \mathcal{M}$ entsprechen dem euklidischen Abstand, zu dessen Berechnung sind für alle Knoten deren Koordinaten als Paare von ganzzahligen Zufallszahlen aus dem Intervall $[0, 500]$ gegeben. [8]

Es wurden für jede Kombination von n, H und Preisklasse der Lagerkosten je zwei verschiedene Testinstanzen geprüft und somit Auswertungen von insgesamt 48 Instanzen durchgeführt. Die Ergebnisse sind in den Tabellen A.1 bis A.4 im Anhang angeführt.

In diesen vier Tabellen kennzeichnet die erste Spalte den Namen der getesteten Instanz, n gibt die Anzahl der darin vorkommenden Kunden an. Die Kosten der optimalen Lösung L_{opt} wurden aus [1] entnommen. Die Spalten 4–6 beziehen sich auf das in Kapitel 3.4 beschriebene exakte Lösungsverfahren mithilfe eines Branch-and-Bound Solvers. L_{ILP} gibt den Wert an, welche das Verfahren zurückgibt. Dabei ist darauf hinzuweisen, dass sich bei gefundener Lösung L_{ILP} minimal von L_{opt} unterscheidet. Diese Abweichung ist auf Rundungsfehler in der Distanzermittlung zurückzuführen, im Fall der hier durchgeführten Implementierungen wurden die euklidischen Distanzen auf zwei Nachkommastellen gerundet. Spalte 5 gibt an, ob der Solver mit dem Branch-and-Bound Verfahren eine Lösung finden konnte. Die jeweilige dafür benötigte Laufzeit zu dessen Bestimmung der Lösung wird in t_{ILP} aufgelistet.

Die letzten drei Spalten nehmen Bezug auf das näherungsweise Lösungsverfahren durch die in Kapitel 3.5 beschriebene Tabusuche. Bei den Tests wurden folgende Parametereinstellungen vorgenommen:

- Erstellte Nachbarlösungen je Operator: $m = 200$
- Länge der Tabuliste: $l = 15$
- Anzahl der durchgeführten Iterationen: $MaxIter = 200$

L_{Tabu} bzw. t_{Tabu} geben die Mittelwerte der zurückgegebenen Funktionswerte bzw. der benötigten Laufzeit aus fünf durchgeführten Testläufen an. Die Spalte „%-Fehler“ zeigt die prozentuelle Abweichung von L_{Tabu} im Vergleich zu L_{opt} auf.

Was die Ergebnisse der ILP Implementierung betrifft, bestätigt sich die Vermutung, dass Probleme nur bis zu einer gewissen Größe lösbar sind. Instanzen mit $n = 5$ und $H = 3$ bzw. $H = 6$ führen zu Problemstellungen mit 102 bzw. 198 Variablen und können in Sekundenschnelle gelöst werden. Erhöht man die Kundenanzahl auf $n = 10$, so liefert das Programm auch noch das exakte Ergebnis, wobei hier aufgrund der erhöhten Anzahl

an Variablen mit 272 bzw. 533 eine erhebliche Steigerung der Laufzeit zu bemerken ist. Ab $n = 15$ stößt das Programm an seine Grenzen, es kann kein Resultat mehr ausgegeben werden.

Betrachtet man die von der Tabusuche erzielten Resultate, lassen sich beträchtliche Unterschiede in der Qualität der Lösungen erkennen, vor allem was die beiden Preiskategorien $[0.1, 0.5]$ und $[0.1, 0.5]$ der Lagerkosten betrifft. In den Tabellen A.1 und A.2 konnten mit durchschnittlichen Abweichungen von 3.09% bzw. 3.49% zur optimalen Lösung durchaus gute Ergebnisse erzielt werden. Bei den insgesamt 24 Testläufen, die mit verschiedenen Instanzen aus der höheren Lagerkostenkategorie durchgeführt wurden, wurde lediglich bei zwei davon eine Lösung gefunden, welche mehr als 5% Abweichung zur optimalen Lösung aufweist.

Ein anderes Bild stellen Ergebnisse bei den Instanzen mit Lagerkosten aus dem Intervall $[0.01, 0.05]$ dar, welche in den Tabellen A.3 und A.4 aufgelistet sind. Hier betragen die durchschnittlichen Abweichungen von der optimalen Lösung 6.68% für $H = 3$ bzw. 8.11% für $H = 6$ und sind somit mehr als doppelt so groß wie jene in der höheren Preisklasse. Ganze 16-mal tritt bei den 24 verschiedenen Testinstanzen dieser Kategorie eine größere Abweichung als 5% auf.

Um eine mögliche Begründung für diese erheblichen Unterschiede in den Testinstanzen zu finden, ist es sinnvoll, die Zielfunktion etwas genauer zu betrachten. Diese setzt sich aus den Lagerkosten von Lieferanten und Kunden sowie den Transportkosten zusammen, siehe Abschnitt 3.5. Die Werte der Instanzen aus den Tabellen A.3 bzw. A.4 und den Tabellen A.1 bzw. A.2 sind ident, mit dem einzigen Unterschied, dass bei den letzten beiden die Lagerkosten nur ein Zehntel der ersten beiden entsprechen. Daher stellen die Lagerkosten in der Zielfunktion nur noch einen wesentlich geringeren Beitrag dar. Anders formuliert: Je geringer die Lagerkosten, umso mehr dominieren die Transportkosten die Zielfunktionswerte. Dadurch wäre es bei solchen Problemstellung von essentieller Bedeutung, neben den richtigen Entscheidungen der Lieferzeitpunkte auch ein besonderes Augenmerk auf die optimale Tourenplanung zu legen.

Genau hier könnte nämlich eine mögliche Schwachstelle in der implementierten Tabusuche vorliegen: Die Reihenfolge der Kundenbelieferungen und somit die Routen an den einzelnen Zeitperioden sind auf dem zufälligen Handeln der Nachbarschaftsoperatoren aufgebaut. Eine Möglichkeit dem entgegenzuwirken wäre, die Implementierung dahingehend zu erweitern, dass je Zeitperiode die zu beliefernden Kunden in der möglichst optimalen Tour abgefahren werden. Bei jeder Erzeugung einer neuen Nachbarlösung müss-

te also ein entsprechender Algorithmus auf diejenige Zeitperiode angewendet werden, wo eine Änderung stattgefunden hat.

Die Problemstellung, bei gegebener Distanzmatrix eine Reihenfolge für eine geplante Rundreise zu finden, sodass die Gesamtdistanz minimal ist, wird bereits seit den 1930er Jahren erforscht. Besser bekannt ist diese Problematik als das *Travelling Salesman Problem*. Lösungen für dieses Problem werden in [4] und [5] beschrieben, würden aber den Umfang dieser Arbeit sprengen.

5. Fazit

Im Gegensatz zu reinen Tourenplanungsproblemen kam das Inventory Routing Problem viel später, erstmals in den 80er bzw. 90er Jahren des vorigen Jahrhunderts, auf. Dabei kann es in den verschiedensten Branchen und in den unterschiedlichsten Ausführungen Anwendung finden. In dieser Arbeit wurde das Single-Product Single-Vehicle Inventory Routing Problem mit der Order-up-to-Level Strategie behandelt. Aus mathematischer Sicht betrachtet, handelt es sich dabei um ein gemischt ganzzahliges lineares Optimierungsproblem.

Eine kombinatorische Möglichkeit ein solches gemischt ganzzahliges lineares Optimierungsproblem exakt zu lösen, stellt das Branch-and-Bound Verfahren dar. Der Nachteil dabei ist jedoch, dass Lösungen nur für Problemstellungen bis zu einer bestimmten Größenordnung gefunden werden können.

In dieser Bachelorarbeit wurde dieses exakte Verfahren mithilfe des Paketes *Google OR Tools* angewendet. Die Grenze der lösbaren Probleme liegt in diesem Fall bei einer Anzahl von zehn Kunden und einem zu betrachtenden Zeitraum von sechs Perioden. In der Praxis wird dies deutlich zu wenig sein, man wird mit signifikant größeren Problemstellungen konfrontiert werden. Um auch dafür Lösungen zu finden, muss der Kompromiss eingegangen werden, nur eine näherungsweise Lösung zu erhalten.

Ein möglicher Ansatz dafür stellt eine Metaheuristik in Form einer Tabusuche dar. Ausgehend von der aktuellen Lösung sucht man in deren Nachbarschaft nach der nächsten. Während dabei auch eine mögliche Verschlechterung erlaubt ist, verbietet man jedoch gewisse Operationen, die in der Tabuliste hinterlegt sind, um Zyklen in der Suche zu vermeiden. Mit der in dieser Arbeit beschriebenen Tabusuche konnten auch größere Instanzen gelöst werden. Jedoch ist ein beträchtlicher Unterschied in der Qualität der Lösungen hinsichtlich unterschiedlicher Preisklassen der Lagerkosten festzustellen. Während die vorgenommene Implementierung bei Problemen mit höheren Lagerkosten mit einer durchschnittlichen Abweichung von ca. 3% eine durchaus gute Performance zeigt, konnte bei den Instanzen mit geringeren Lagerkosten eine mehr als doppelt so große mittlere

Abweichung festgestellt werden. Eine Hypothese für diesen größeren Fehler bei zweiteren Instanzen ist eine fehlende intelligente Tourenplanung, um die dort dominierenden Transportkosten möglichst gering zu halten.

Der Ausblick auf weitere Schritte zur besseren Handhabung des Inventory Routing Problems richtet sich also speziell auf das Teilproblem, optimale Routen für die vorzunehmenden Auslieferungen zu finden. Darauffolgend kann man Erweiterungen des Modells im Sinne von mehreren verfügbaren Lieferfahrzeugen oder mehreren zu betrachtenden Produkten vornehmen, welche die Komplexität der Problemstellung nochmals deutlich erhöhen werden.

Literatur

[1] Leandro C. Coelho. *Prof. Leandro C. Coehlo, PH.D. - Inventory Routing*. Zugriff am: 07.2021. URL: https://www.leandro-coelho.com/instances/inventory-routing/.

[2] Leandro C. Coelho und Jean-François Cordeau und Gilbert Laporte. *Thirty Years of Inventory Routing*. In: *Transportation Science* 48 (2014), S. 1–19. DOI: 10.1287/trsc.2013.0472.

[3] Christos Papadimitriou und Kenneth Steiglitz. *Combinatorial Optimization: Algorithms and Complexity*. Bd. 32. Jan. 1982. ISBN: 0-13-152462-3. DOI: 10.1109/TASSP.1984.1164450.

[4] Shen Lin. *Computer solutions of the traveling salesman problem*. In: *The Bell System Technical Journal* 44.10 (1965), S. 2245–2269. DOI: 10.1002/j.1538-7305.1965.tb04146.x.

[5] Shen Lin und Brian W. Kernighan. *An Effective Heuristic Algorithm for the Traveling-Salesman Problem*. In: *Oper. Res.* 21 (1973), S. 498–516.

[6] Walter J. Bell und Louis M. Dalberto und Marshall L. Fisher und Arnold J. Greenfield und R. Jaikumar und Pradeep Kedia und Robert G. Mack und Paul J. Prutzman. *Improving the Distribution of Industrial Gases with an On-Line Computerized Routing and Scheduling Optimizer*. In: *Interfaces* 13 (1983), S. 4–23.

[7] Dr. Marco Lübbecke. *Wirtschaftslexikon24 - Branch-and-Bound-Verfahren*. Zugriff am: 03.2021. URL: https://wirtschaftslexikon.gabler.de/definition/branch-and-bound-verfahren-28551.

[8] Claudia Archetti und Luca Bertazzi und Alain Hertz und M. Grazia Speranza. *A Hybrid Heuristic for an Inventory Routing Problem*. In: *INFORMS Journal on Computing* 24 (2012). DOI: 10.1287/ijoc.1100.0439.

[9] Claudia Archetti und Luca Bertazzi und Gilbert Laporte und M. Grazia Speranza. *A Branch-and-Cut Algorithm for a Vendor-Managed Inventory-Routing Problem*. In: *Transportation Science* 41 (2007), S. 382–391. DOI: 10.1287/trsc.1060.0188.

[10] Luca Bertazzi und M. Grazia Speranza. *Inventory routing Problems: an introduction*. In: *EURO Journal on Transportation and Logistics* (2012), S. 307–326. DOI: 10.1007/s13676-012-0016-7.

[11] Fred Glover und Manuel Laguna und Rafael Marti. *Tabu Search*. Bd. 16. Juli 2008. DOI: 10.1007/978-1-4615-6089-0.

[12] Sherif Shokry und Shinji Tanaka und Fumihiko Nakamura und Ryo Ariyoshi und Shino Miura. *Bandwidth Maximization Approach for Displaced Left-Turn Crossovers Coordination under Heterogeneous Traffic Conditions*. In: *Journal of Traffic and Transportation Engineering* 6 (Sep. 2018), S. 183–196. DOI: 10.17265/2328-2142/2018.04.004.

[13] *Wirtschaftslexikon24 - Vendor Managed Inventory (VMI)*. Zugriff am: 03.2021. URL: http://www.wirtschaftslexikon24.com/d/vendor-managed-inventory-vmi/vendor-managed-inventory-vmi.htm.

A. Auswertungen

Tabelle A.1.: Rechenergebnisse auf Instanzen mit $H = 3$ und $h_i \in [0.1, 0.5]$

Instanz	n	L_{opt}	L_{ILP}	Erfolg	t_{ILP}	L_{Tabu}	t_{Tabu}	%-Fehler
abs1n5.dat	5	2149.80	2149.65	ja	0.1	2253.87	20	4.48
abs3n5.dat	5	3265.44	3267.10	ja	1	3488.17	18	6.82
abs2n10.dat	10	4803.17	4802.81	ja	36	4802.81	29	0.00_1
abs4n10.dat	10	4347.06	4348.02	ja	62	4348.02	29	0.00_1
abs3n15.dat	15	6711.25	-	nein	-	6894.41	73	2.73
abs5n15.dat	15	5210.85	-	nein	-	5425.22	44	4.11
abs4n20.dat	20	7050.91	-	nein	-	7665.39	110	8.71
abs1n20.dat	20	7353.82	-	nein	-	7607.98	60	3.46
abs5n25.dat	25	10857.68	-	nein	-	10918.55	163	0.56
abs2n25.dat	25	9266.87	-	nein	-	9440.24	219	1.18
abs1n30.dat	30	12635.55	-	nein	-	12782.02	216	1.16
abs3n30.dat	30	12509.26	-	nein	-	12942.46	381	3.46
								⌀ 3.09

Tabelle A.2.: Rechenergebnisse auf Instanzen mit $H = 6$ und $h_i \in [0.1, 0.5]$

Instanz	n	L_{opt}	L_{ILP}	Erfolg	t_{ILP}	L_{Tabu}	t_{Tabu}	%-Fehler
abs1n5.dat	5	5942.82	5942.73	ja	6	6187.51	82	4.12
abs3n5.dat	5	6956.28	6957.47	ja	4	6971.18	43	0.21
abs2n10.dat	10	8569.73	8569.88	ja	2010	8982.45	85	4.82
abs4n10.dat	10	8792.29	8797.42	ja	1701	9174.54	159	4.35
abs3n15.dat	15	13554.15	-	nein	-	13770.32	465	1.59
abs5n15.dat	15	10385.54	-	nein	-	10766.74	113	3.67
abs4n20.dat	20	14539.72	-	nein	-	15249.86	475	4.88
abs1n20.dat	20	14702.95	-	nein	-	15425.09	446	4.91
abs5n25.dat	25	19047.70	-	nein	-	19570.16	998	2.74
abs2n25.dat	25	16823.16	-	nein	-	17385.18	1093	3.34
abs1n30.dat	30	23183.99	-	nein	-	24044.50	681	3.71
abs3n30.dat	30	23382.73	-	nein	-	24200.29	922	3.50
								⌀ 3.49

Tabelle A.3.: Rechenergebnisse auf Instanzen mit $H = 3$ und $h_i \in [0.01, 0.05]$

Instanz	n	L_{opt}	L_{ILP}	Erfolg	t_{ILP}	L_{Tabu}	t_{Tabu}	%-Fehler
abs1n5.dat	5	1281.68	1281.53	ja	0.2	1383.59	24	7.95
abs3n5.dat	5	2020.65	2022.31	ja	0.3	2257.94	18	11.74
abs2n10.dat	10	2510.13	2509.77	ja	105	2509.77	32	0.00[1]
abs4n10.dat	10	2188.01	2188.97	ja	56	2188.97	30	0.00[1]
abs3n15.dat	15	2841.06	-	nein	-	3050.23	86	7.36
abs5n15.dat	15	2453.50	-	nein	-	2685.88	50	9.47
abs4n20.dat	20	3239.31	-	nein	-	3836.34	118	18.43
abs1n20.dat	20	2793.29	-	nein	-	2971.51	57	6.38
abs5n25.dat	25	3695.94	-	nein	-	3736.52	141	1.10
abs2n25.dat	25	3495.97	-	nein	-	3570.99	166	2.15
abs1n30.dat	30	3918.76	-	nein	-	4056.81	156	3.52
abs3n30.dat	30	3761.85	-	nein	-	4217.36	206	12.11
								⌀ 6.68

Tabelle A.4.: Rechenergebnisse auf Instanzen mit $H = 6$ und $h_i \in [0.01, 0.05]$

Instanz	n	L_{opt}	L_{ILP}	Erfolg	t_{ILP}	L_{Tabu}	t_{Tabu}	%-Fehler
abs1n5.dat	5	3335.24	3335.15	ja	3	3684.89	39	10.48
abs3n5.dat	5	4776.00	4777.19	ja	5	4784.84	44	0.19
abs2n10.dat	10	5236.98	5237.00	ja	1743	5825.40	313	11.24
abs4n10.dat	10	5104.91	5109.60	ja	965	5457.49	196	6.91
abs3n15.dat	15	6060.38	-	nein	-	6323.95	479	4.35
abs5n15.dat	15	5309.48	-	nein	-	5419.77	119	2.08
abs4n20.dat	20	7432.78	-	nein	-	8222.86	595	10.63
abs1n20.dat	20	6490.18	-	nein	-	7021.78	671	8.19
abs5n25.dat	25	7452.28	-	nein	-	8100.08	559	8.69
abs2n25.dat	25	7484.84	-	nein	-	8232.95	856	9.99
abs1n30.dat	30	8319.59	-	nein	-	9361.91	1107	12.53
abs3n30.dat	30	8214.55	-	nein	-	9200.95	1135	12.01
								⌀ 8.11

[1] %-Fehler wurde auf 0.00 gesetzt, da optimale Lösung gefunden wurde. Abweichung resultiert aus Rundungsfehler in der Transportkostenberechnung.