FAKULTÄT
FÜR !NFORMATIK
Faculty of Informatics

# Optimierung der periodischen Tourenplanung in der Müllentsorgung

DIPLOMARBEIT

zur Erlangung des akademischen Grades

**Diplom-Ingenieur**

im Rahmen des Studiums

**Computational Intelligence**

eingereicht von

**Karl-Michael Edlinger, BSc.**
Matrikelnummer 0326159

an der
Fakultät für Informatik der Technischen Universität Wien

Betreuung: Univ.-Prof. Dipl.-Ing. Dr.techn. Günther Raidl
Mitwirkung: Univ.-Prof. Dr. Karl Dörner,
Dr. Günter Kiechle

Wien, TT.MM.JJJJ

(Unterschrift Verfasser) (Unterschrift Betreuung)

# Erklärung zur Verfassung der Arbeit

Karl-Michael Edlinger, BSc.
Schwadorf 6, 3100 Sankt Pölten

Hiermit erkläre ich, dass ich diese Arbeit selbständig verfasst habe, dass ich die verwendeten Quellen und Hilfsmittel vollständig angegeben habe und dass ich die Stellen der Arbeit - einschließlich Tabellen, Karten und Abbildungen -, die anderen Werken oder dem Internet im Wortlaut oder dem Sinn nach entnommen sind, auf jeden Fall unter Angabe der Quelle als Entlehnung kenntlich gemacht habe.

| | |
|---|---|
| (Ort, Datum) | (Unterschrift Verfasser) |

# Danksagung

Ich danke meinen Betreuern, Prof. Günther Raidl, Prof. Karl Dörner und Günter Kiechle, für ihr konstruktives Feedback und ihre Ideen, welche mir ermöglicht haben, immer neue Aspekte der Problemstellung zu erkennen.

Horst Stadler und Fritz Payr, meinen Kollegen bei Salzburg Research, danke ich für ihre Unterstützung bei dem Projekt und die interessanten Gespräche über Schnittstellen und Datenstrukturen. In zahlreichen Diskussionen hat mir Johannes Strodl mit seinem Fachwissen geholfen, ein besseres Verständnis für die theoretische Problemstellung und für die Lösungsverfahren zu gewinnen.

Besonderer Dank gilt auch meinen Eltern, Elfriede und Karl Edlinger, die mich während meines Studiums und all meiner Vorhaben immer unterstützt haben.

# Abstract

The volume of waste is rising in Austria - especially private households are producing more and more garbage. It is quite a difficult logistical problem to bring all the garbage to recycling centers and land fills. In Vienna alone, 100.000 garbage bins are emptied every day. 270 compactor trucks bring the garbage to land fills. To plan the waste collection is a challenge for many cities. A lot of time and know-how is invested in the planing process, still the solutions are not always satisfactory. In the literature there are already some solution procedures for waste collection, but none are viable in practice. This thesis tries to answer the question if it is possible to support the planning process for periodic waste collection to reduce the cost of operations.

A plan is created not just for one day, but for several, e.g. one week. After this period of time the plan is repeated. In the planing period some garbage bins have to be emptied more than once. Visiting schedules can be derived from this frequency. A visiting schedule states on which days of the planing period a garbage bin is emptied. Once each site has been assigned one visiting schedule, we can create tours for the compactor trucks. A tour is a sequence of sites, and each tour is assigned to one vehicle. Each truck starts its daily tour from its depot. There may be several depots from which the vehicles depart. On its tour, the vehicle will empty garbage bins. Before the vehicle reaches its maximum capacity, it will drive to a land fill and unload the garbage. Land fills, recycling centers, and incinerating facilities are summarized under the term *Intermediate Facilities (IF)*. For the planner, there are several IFs to choose from. Not always is the nearest IF the best choice and in some cases policies mandate that a certain IF is used or preferred. At the end of the tour the vehicle returns to its depot. A specialty is that a truck is also allowed to return to the depot partially loaded and drive to an IF on the next day. The literature classifies the basic problem as a *Multi-Depot Periodic Vehicle Routing Problem with Intermediate Facilites (MDPVRP-IF)*.

The MDPVRP-IF with the described extensions requires a flexible solving method. The *Variable Neighborhood Search (VNS)* has already proved its ability to find good solutions for similar problems and has been adopted for this problem. As local search we use 2-opt. The acceptance criteria has been enriched with principles from *Simulated Annealing (SA)*. The VNS heuristic has been tested with real data from an Austrian waste collection company. The solving procedure could solve all test instances in under 30 minutes. The dispatcher of the company that provided the data stated that all solutions show an improvement over the existing solutions. Thus the company intends to use the solution procedure in the next planning process.

# Kurzfassung

Das Müllaufkommen in Österreich steigt stetig an, auch das der Privathaushalte. Den produzierten Müll zu entsorgen, stellt mittlerweile ein großes logistisches Problem dar. So werden in Wien täglich an die 100.000 Müllbehälter entleert. 270 Fahrzeuge bringen den Müll zu Mülldeponien. Die Planung der periodischen Entleerung von Abfallbehältern stellt für viele Städte eine Herausforderung dar. Viel Knowhow und Zeit werden in die Planung investiert, und trotzdem werden nicht immer zufriedenstellende Lösungen gefunden. In der Fachliteratur existieren zwar einige Lösungsverfahren für die Müllsammlung, doch keines, das auch die vielen Voraussetzungen für den Praxiseinsatz erfüllen würde. Sollte es aber nicht doch möglich sein, die Planungsbeauftragten bei ihren Aufgaben zu unterstützen und dadurch die Kosten zu reduzieren?

Genauer betrachtet, handelt es sich bei dem Problem der periodischen Entleerung von Abfallbehältern bei Müllsammelstellen um ein *Multi-Depot Periodic Vehicle Routing Problem with Intermediate Facilites (MDPVRP-IF)*. Das bedeutet, dass ein oder mehrere Fahrzeuge von verschiedenen Depots aus losfahren und Müllbehälter entleeren. Dies passiert über eine längere Planungsperiode (z.B.: eine Arbeitswoche). In diesem Planungszeitraum müssen einige Behälter mehrmals entleert werden. Aus der Anzahl der notwendigen Entleerungen im Planungszeitraum ergeben sich Besuchsmuster. Ein solches Besuchsmuster gibt an, an welchen Tagen ein Abfallbehälter entleert wird und an welchen nicht. Für jeden Müllcontainer muss ein Besuchsmuster ausgewählt werden. Daraus ergibt sich für die Fahrzeuge jeden Tag eine Liste an Behältern, die sie entleeren. Hat ein Fahrzeug einige Müllbehälter entleert und seine maximale Ladekapazität fast erreicht, so muss das Fahrzeug zu einer Entladestation fahren, auch *Intermediate Facility (IF)* genannt. Eine Besonderheit der Müllsammlung ist, dass die Fahrzeuge unter bestimmten Umständen auch beladen ins Depot fahren dürfen und erst am nächsten Tag zu einer IF. Eine weitere Eigenart besteht in der Auswahl der IFs. Durch Verträge mit IFs kann es notwendig sein, dass nicht zur nächstgelegenen IF gefahren wird, sondern zu einer anderen.

Diese Erweiterungen des MDPVRP-IF verlangen ein flexibles Lösungsverfahren. Das *Variable Neighborhood Search-Verfahren (VNS)* hat schon bei ähnlichen Problemen seine Vorzüge bewiesen und wurde in der vorliegenden Arbeit für dieses Problem adaptiert. Als lokale Suche kommt *2-Opt* zum Einsatz und als Akzeptanzkriterium wird *Simulated Annealing (SA)* verwendet. Getestet wurde das VNS mit Echtdaten eines österreichischen Entsorgungsunternehmens. Die Ergebnisse sind vielversprechend: In weniger als 30 Minuten lieferte das Verfahren für alle Testszenarien Lösungen, die von einem Disponenten als Verbesserungen im Vergleich zu den bisherigen Lösungen bezeichnet wurden. Ein Einsatz des Lösungsverfahrens in der nächsten Planungsperiode ist angedacht.

# Inhaltsverzeichnis

# Abbildungsverzeichnis

# Liste der Algorithmen



# Tabellenverzeichnis

KAPITEL 1

# Einleitung

Es gibt immer mehr Menschen auf dieser Welt und alle produzieren Müll. Abfall, Mist und Müll aber sind dasselbe, Gegenstände, die nicht mehr gebraucht werden, aber Platz verbrauchen und noch andere negative Eigenschaften besitzen. Daher wird Müll gesammelt und aus den Lebensräumen der Menschen gebracht.

Einige Abfallsorten werden wiederverwertet, der meiste Müll wird jedoch auf Mülldeponien gelagert oder in Müllverbrennungsanlagen in Wärme und Energie umgewandelt.

Das Problem dabei ist, dass die Standorte der Entsorgungsanlagen nicht den Entstehungsorten von Müll entsprechen. Der Abfall muss erst zu einer Verarbeitungsanlage gebracht werden. In Österreich werden dazu von der öffentlichen Verwaltung Abfallsammelbehälter aufgestellt und in regelmäßigen Abständen entleert.

Um Müll korrekt entsorgen zu können, muss er getrennt werden. Speziell Glas, Kunststoff, Metall und Papier werden in vielen Teilen Österreichs vom restlichen Hausmüll getrennt gesammelt. Da die Mengen dieser Abfallsorten geringer sind, bekommt nicht jedes Wohnhaus einen eigenen Sammelbehälter, sondern ein Behälter wird an einem öffentlich zugänglichen Ort für mehrere Wohnhäuser aufgestellt. Wenn unterschiedliche Müllsorten an einem Ort gesammelt werden, wird dies als Müllsammelstelle bezeichnet.

Die Abfallbehälter von Müllsammelstellen müssen in regelmäßigen Abständen entleert werden. Die Häufigkeit ist jedoch für jede Müllsammelstelle unterschiedlich. Abhängig davon, wie viele Menschen ihren Abfall zu einer Müllsammelstelle bringen, kann ein mehrmaliges Entleeren pro Woche notwendig sein. Wird eine Sammelstelle seltener genutzt, reicht einmal im Monat aus.

Für die Abfallwirtschaft ist es wichtig, den Müll von den Müllsammelstellen zu den Verarbeitungsanlagen und Lagerungsstätten zu transportieren, und das möglichst kostengünstig.

Ein weiterer wichtiger Faktor für die Abfallwirtschaft ist die Zufriedenheit der Bewohner, von denen der Müll abgeholt wird. Diese Menschen sind zufrieden, wenn der Abfall in regelmäßigen Abständen abgeholt wird und die Abfallcontainer nicht überquellen. Beschwerden gibt es vor allem, wenn die Müllabfuhr in den frühen Morgenstunden kommt. Denn das Entleeren der Müllcontainer kann sehr laut sein und viele Anrainer von Müllsammelstellen sind verärgert,

wenn sie frühzeitig geweckt werden. In der Nähe von Schulen ist es sogar untersagt, dass während der Unterrichtszeiten Müllcontainer entleert werden. Die Abfallwirtschaft ist daher bemüht, die Zeiten, zu denen Abfallcontainer entleert werden, den Wünschen der Anrainer anzupassen.

Eine weitere Herausforderung besteht darin, dass nicht immer gleich viel Müll entsteht. Im Sommer, wenn weniger Menschen zuhause sind, fällt weniger Abfall an. Die Weihnachtsfeiertage sind das andere Extrem. Hier müssen zusätzliche Touren gefahren werden, um Müllberge neben den Abfallcontainern zu vermeiden.

Touren für die Müllsammelfahrzeuge werden für eine längere Planungsperiode zusammengestellt. Durch die Planungsperiode ist gegeben, nach wie vielen Tagen sich ein Plan zum ersten Mal wiederholt. Im Falle einer Planungsperiode von 14 Tagen existieren 14-Tages-Touren für jedes Fahrzeug. Diese Touren werden wiederholt gefahren, bis es zu einer Neuplanung kommt.

Die Planung von Touren für eine längere Planungsperiode erfordert viel Knowhow und Zeit. Planer, die das notwendige Wissen und auch ausreichend Erfahrung haben, sind gefragt. Kleine und mittlere Unternehmen können es sich meist nicht leisten, einen eigenen Planer anzustellen, und greifen auf die Dienste von externen Beratern zurück. Die Arbeit wird dadurch aber nicht einfacher und kann auch nicht schneller erledigt werden.

Durch die Unterstützung von Computern mit spezieller Software soll die Planung von Touren einfacher werden und so die notwendige Zeit zum Planen reduziert werden. Auch erhofft man sich Einsparungen bei der Fahrzeit, eine bessere Ausnutzung der vorhandenen Ressourcen und eine bessere Routenführung.

Der Gegenstand dieser Arbeit ist nun die Entwicklung eines Verfahrens zur Erstellung und Optimierung von Routen für mehrere Müll-Fahrzeuge über einen längeren Planungshorizont.

KAPITEL 2

# Periodic Vehicle Routing Problem

## 2.1 Definition

Die regelmäßige Müllabholung und die in wiederkehrenden Abständen notwendige Auslieferung von Gütern haben zur Definition des *Periodic Vehicle Routing Problem (PVRP)* geführt. Bei dieser Art von Problem müssen bestimmte Orte oder Kunden in einem Planungszeitraum von mehreren Tagen mindestens einmal beliefert werden. Der Planungszeitraum kann einen oder mehrere Tage betragen. Wird nur für einen Tag geplant und stehen mehrere Fahrzeuge zur Verfügung, so handelt es sich um ein *Vehicle Routing Problem (VRP)*. Eine Flotte von Fahrzeugen mit begrenzter Kapazität beliefert die Kunden von einem zentralem Depot aus. Steht nur ein Fahrzeug zur Verfügung, spricht man von einem *Travelling Salesman Problem (TSP)*. Besteht der Planungszeitraum aus mehreren Tagen, wird es als *Periodic Travelling Salesman Problem (PTSP)* klassifiziert.

Es muss sich dabei aber nicht immer um eine Auslieferung von Gütern handeln. In vielen Fällen geht es um eine Abholung von Gütern beim Kunden. Die Müllsammlung ist nur einer dieser Fälle. Im weiteren Verlauf dieses Kapitels wird davon ausgegangen, dass Kunden Güter anbieten, welche von Fahrzeugen abgeholt werden sollen.

Im Planungszeitraum muss jeder Kunde zumindest einmal besucht werden, um dessen Güter abzuholen. Sollte das Angebot des Kunden größer sein, so ist es notwendig, den Kunden öfter zu besuchen. Wie oft ein Kunde zu besuchen ist, gibt die Besuchsfrequenz an, z.B. täglich oder zweimal in der Woche. Wird für eine Arbeitswoche geplant und einige Kunden müssen nur ein bis viermal beliefert werden, so gibt es für jeden dieser Kunden mehrere Möglichkeiten, wann ein Fahrzeug den Kunden besuchen kann, z.B. für zweimal in der Woche ({Montag, Dienstag}, {Montag, Mittwoch}, {Montag, Donnerstag}, {Montag, Freitag}, {Dienstag, Mittwoch}, {Dienstag, Donnerstag} ...). Diese Kombinationen werden Besuchsmuster genannt. Besuchsmuster ergeben sich aus der Länge des Planungszeitraums, der Besuchsfrequenz und zusätzlichen Einschränkungen. So ist es in vielen Fällen unerwünscht, an zwei aufeinanderfolgenden Tagen besucht zu werden. Bei der Abholung von Abfall fällt in der kurzen Zeit wahrscheinlich nicht genug Müll an, um einen Abfallcontainer zu füllen, aber in der restlichen Zeit bis zur

nächsten Abholung kann mehr Müll entstehen, als in einen Container passt. Bei der Belieferung eines Restaurants mit Lebensmitteln ist es zum Beispiel nicht immer möglich, bei einer Besuchsfrequenz von zweimal in der Woche die Lebensmittel am Montag und Dienstag zu bringen. Erstens ist vielleicht nicht der notwendige Lagerraum vorhanden, und zweitens halten sich einige Lebensmittel nicht lange genug. Aus dem vorherigen Beispiel für Besuchsmuster wären Kombinationen wie {Montag, Dienstag} daher nicht Teil der möglichen Besuchsmuster.

Abbildung 2.1 zeigt eine beispielhafte Lösung eines PVRP mit zehn Kunden, einem Planungshorizont von einer Arbeitswoche (fünf Tage), einem Depot und zwei Fahrzeugen. Für die Kunden existieren unterschiedliche Besuchsmuster. Kunden, die durch einen schwarzen Kreis dargestellt sind, müssen fünfmal in fünf Tagen besucht werden. Zu jenen Kunden, die weiß ausgemalt sind, soll in dem Planungszeitraum zweimal gefahren werden. Aus den Besuchsfrequenzen wurden bereits Besuchsmuster generiert und ausgewählt. Für das Fahrzeug 1 ergeben sich dadurch zwei Routen, welche abwechselnd gefahren werden: je eine am Montag, Mittwoch und Freitag. Bei dieser Route werden alle Kunden mit einer Besuchsfrequenz von fünf besucht. Die Route, die am Dienstag und Donnerstag gefahren wird, besucht jene Kunden mit der Besuchsfrequenz von zwei. Das Fahrzeug 2 ist nur am Dienstag und Donnerstag im Einsatz.

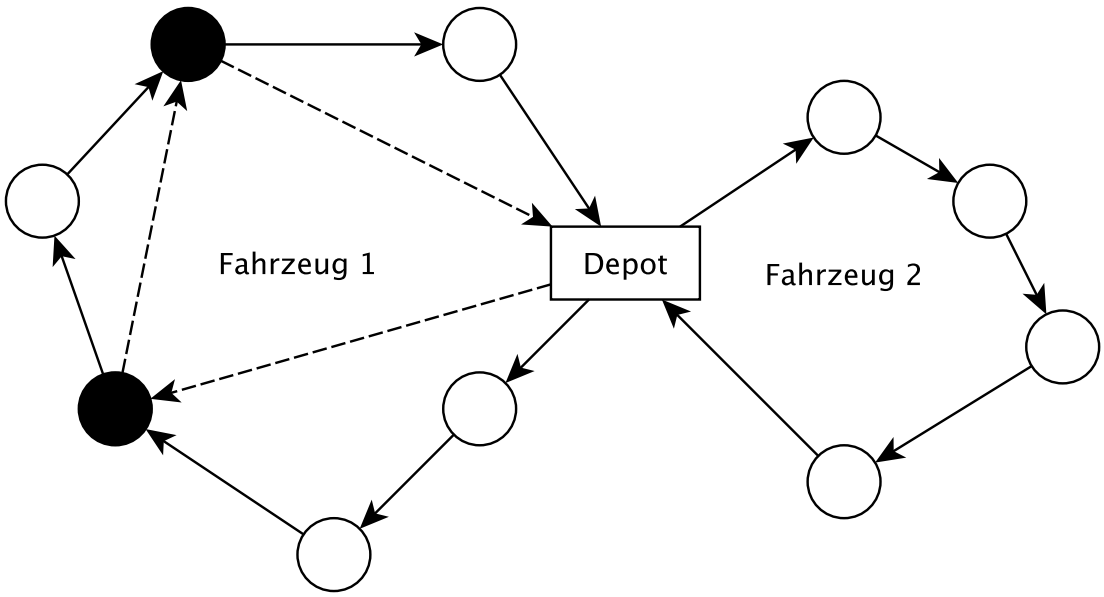



**Abbildung 2.1:** Ein Periodic Vehicle Routing Problem (PVRP) mit zehn Kunden, einem Planungshorizont von fünf Tagen und zwei Fahrzeugen. Die Kunden mit einem schwarzen Kreis sind fünfmal zu besuchen, alle anderen sind zweimal zu besuchen.

Für jeden Tag des Planungszeitraums ist eine Aufteilung der zu besuchenden Kunden auf die zur Verfügung stehenden Fahrzeuge notwendig. Alle Fahrzeuge starten von einem Depot und kehren am Ende des Tages auch zu diesem Depot wieder zurück. Im klassischem PVRP wird davon ausgegangen, dass der Arbeitstag für dieses Fahrzeug zu Ende ist, sobald das Fahrzeug

zum Depot zurückkehrt. Ein Auffüllen der Ladung oder eine Entleerung, je nach gegebenem Problem, ist nicht vorgesehen.

Jedes Fahrzeug hat eine Anzahl von Kunden, die es zu besuchen hat. Dazu kommt noch eine beschränkte Ladekapazität. In den meisten Fällen aus der Praxis kommt auch noch eine Begrenzung der maximalen Fahrzeit hinzu. Diese hängt oft mit der rechtlich oder gewerkschaftlich festgelegten maximalen Lenkzeit der Fahrer/innen oder der maximalen Arbeitszeit der Arbeiter/innen zusammen. Es kann aber auch operative Gründe geben, wie Fahrzeuge, welche nur für eine bestimmte Zeit zur Verfügung stehen oder nur zu bestimmten Zeiten fahren dürfen. Für jedes Fahrzeug muss also eine Tour gefunden werden, sodass alle Kunden besucht werden und die Beschränkungen eingehalten werden.

Zusammengefasst sind beim PVRP drei Schritte zu tun, um eine Lösung zu erhalten [6]:

1. Für jeden Kunden ein geeignetes Besuchsmuster wählen.
2. Jedem Fahrzeug für jeden Tag im Planungszeitraum eine Anzahl von Kunden zuweisen.
3. Planen, in welcher Reihenfolge jedes Fahrzeug die zugewiesenen Kunden besucht.

Dabei ist das Ziel, die entstehenden Kosten niedrig zu halten, alle Kunden gemäß ihrer Besuchsmuster zu besuchen und keine Nebenbedingungen zu verletzen.

Das PVRP ist in vielen Bereichen der modernen Gesellschaft zu finden. Das erste Mal wurde es von E. J. Beltrami und L. D. Bodin 1974 in ihrer Arbeit „Networks and Vehicle Routing for Municipal Waste Collection“ beschrieben und algorithmisch gelöst. R. Russel und W. Igo schreiben 1979, auch zum Thema der Müllsammlung [7], über die Schwierigkeit, das richtige Besuchsmuster zu wählen. Daher lautet auch der Name der Arbeit „Assignment Routing Problem“. Die erste formale Definition des PVRP kommt erst 1984 von N. Christofides und J. E. Beasley. Sie betrachten das Problem aus der Perspektive der Routenkonstruktion der Fahrzeuge. Für jeden Tag im Planungszeitraum werden die Touren der Fahrzeuge so erstellt, dass am Ende die Besuchsfrequenzen für alle Kunden eingehalten werden. Auf Grund der Komplexität des Problems versuchten sie allerdings nicht, das Problem optimal zu lösen.

Seit diesen Anfängen wurde das PVRP mit Heuristiken und Metaheuristiken, aber auch mit exakten Verfahren gelöst. Es kamen Erweiterungen für mehrere Depots oder Zeitfenster hinzu. Auch in vielen Bereichen abseits der Müllsammlungfand es Anwendung, z.B. bei der Auslieferung von Wäsche in Krankenhäusern, der Wartung von Aufzügen, der Auslieferung von Blutkonserven [8] und vielen mehr.

## 2.2 Erweiterungen des Periodic Vehicle Routing Problem für die Müllsammlung

Das hier zu modellierende Problem der Tourenfindung für die Abfallsammlung in Österreich unterliegt vielen Regelungen und hat mehr und komplexere Nebenbedingungen als das klassische PVRP. In diesem Kapitel werden nun die Spezialitäten des PVRP für die Müllsammlung beschrieben.

In Österreich sind unterschiedliche Typen von Lastkraftwagen (LKW) zur Leerung von Müllbehältern unterwegs. Sie alle benötigen neben dem Fahrer noch extra Personal, um die Müllbehälter zu entleeren. Fahrzeuge mit Vorrichtungen zum maschinellen Entleeren von Abfallcontainern sind in Österreich derzeit nicht im Einsatz. Die eingesetzten Fahrzeuge unterscheiden sich in der maximalen Kapazität und dadurch in der Bauweise. Im Allgemeinen werden die LKWs nach ihrem Fassungsvolumen in zwei Klassen eingeteilt. Die größeren LKWs haben meistens drei oder mehr Achsen. Für diese LKWs sind nicht alle Straßen befahrbar. Sind Straßen zu eng oder existieren Fahrverbote für die großen LKWs, ist der Einsatz der kleineren LKWs notwendig. Da die größeren LKWs mehr Container entleeren können, bevor sie selbst zum Entleeren fahren müssen, ist ihr Einsatz, wo möglich, wirtschaftlich sinnvoll. Im Folgenden werden die unterschiedliche Kapazitäten von Müllsammelfahrzeugen Beachtung finden. Auf Grund fehlender Daten werden Fahrverbote und Einschränkungen der Befahrbarkeit von Straßen nicht berücksichtigt.

Das Straßennetz in Österreich hat neben Fahrverboten und baulichen Einschränkungen - wie Unterführungen mit niedriger Höhe - noch viele andere Ausnahmen. Dadurch kann fast immer davon ausgegangen werden, dass die Distanzen nicht symmetrisch sind, d.h. dass, um von Punkt $a$ nach Punkt $b$ zu gelangen, eine andere Strecke gefahren werden muss, als von Punkt $b$ zu Punkt $a$. Dadurch ergeben sich unterschiedliche Fahrzeiten und das Berechnen der Streckenlänge und Fahrdauer ist aufwändiger.

Generell gilt, dass die Müllsammelfahrzeuge jegliche Müllsorte laden können. Ein Wechsel, z.B. von Bio-Müll auf Papier, ist zwar durchaus möglich, erfordert jedoch eine Zwischenreinigung. In der Praxis wird daher versucht, so selten wie möglich Müllsorten für ein Fahrzeug zu wechseln. So wird etwa in der erste Hälfte der Woche Metall eingeholt und in der zweiten Hälfte dann Restmüll.

Die Planung findet für jede Müllsorte getrennt statt. Dadurch ist vorgegeben, an welchem Tag ein Fahrzeug für wie viele Stunden zur Verfügung steht. Für das zu entwickelnde Verfahren ergibt sich die Situation, dass nicht alle Fahrzeuge jeden Tag zur Verfügung stehen und die Einsatzzeiten der Fahrzeuge auch unterschiedlich sein können.

Je nach Größe des Entsorgungsunternehmens stehen ein oder mehrere Müllsammelfahrzeuge zur Verfügung. Wenn diese Fahrzeuge nicht im Einsatz sind, stehen sie in einem Depot. Große Unternehmen besitzen unter Umständen mehrere Depots, bei kleineren Unternehmen mag es sogar üblich sein, dass die Fahrzeuge bei den Lenkern zu Hause stehen. Wo auch immer das Fahrzeug abgestellt wird, in weiterer Folge wird dieser Ort als Depot bezeichnet. Jedes Fahrzeug ist genau nur einem Depot zugeordnet und beginnt und beendet seine Touren in diesem Depot.

Die Fahrzeuge starten ihre Touren von ihrem Depot aus, fahren anschließend zu Müllsammelstellen und entleeren dort die Abfallbehälter einer Müllsorte. Neben der Fahrzeit zu den Sammelstellen sind auch die Servicezeiten bei den Sammelstellen für die Tourlänge ausschlaggebend. Unter Servicezeit wird die Zeit verstanden, die es braucht, um nach Eintreffen des Fahrzeugs die Müllcontainer zum Fahrzeug zu bringen, die Container zu entleeren und dann den Container wieder zurückzubringen. Die Servicezeiten hängen von der Anzahl der Müllsammelbehälter, deren Größe und Aufstellungsort ab und schwanken zwischen 30 Sekunden und ein paar Minuten.

Einige Müllsammelstellen dürfen allerdings nur zu bestimmten Zeiten entleert werden. In

der Praxis dürfen diese Müllsammelstellen nur vor Schulbeginn und/oder bevor die Geschäfte öffnen, angefahren werden.

Ist die maximale Ladekapazität eines Fahrzeugs erreicht, so muss es zu einer Mülldeponie, Müllverbrennungsanlage oder einem Zwischenlager fahren, um dort den Müll abzuladen. In der Literatur werden diese Entsorgungsanlagen unter dem Begriff *Intermediate Facility (IF)* zusammengefasst. Für die Entsorgungsunternehmen ist die Art der Entsorgungsanlage uninteressant, weswegen im Folgenden alle Entsorgungsanlagen gleichermaßen als IF bezeichnet werden.

Als Servicezeit bei einer IF wird die Zeit verstanden, die es benötigt um den LKW zu leeren.

Es gibt meistens mehrere IFs zur Auswahl und daher auch bestimmte Modi, nach denen entschieden wird, zu welcher IF der Müll gebracht wird. Bei der freien Zuordnung gibt es keine Einschränkungen und es wird die IF gewählt, welche die Fahrzeit am wenigsten verlängert. Eine prozentmäßige Zuteilung kommt zum Einsatz, wenn Abkommen über die gelieferten Müllmengen zwischen den Betreibern der LKW-Flotte und den IFs existieren. Es ist dann so eine Zuteilung zu den IFs zu treffen, dass die abgelieferte Müllmenge möglichst genau dem abgemachten Prozentsatz des gesamten Müllvolumens entspricht. Wird Abfall auch von Gewerbebetrieben abgeholt, kann es oft notwendig sein, dass dieser zu einer bestimmten IFs gebracht wird. Dies kann entsorgungstechnische Gründe haben oder mit Verträgen zwischen den Gewerbebetrieben und IFs zu tun haben. Dieser Modus tritt auch oft in Kombination mit der freien oder prozentmäßigen Zuordnung auf. Nur selten ist eine IF für alle Sammelorte vorgegeben.

Am Ende der Tour fährt das Müllsammelfahrzeug wieder in das Depot, von dem es gestartet ist, zurück. Das Depot ist dabei keine IF, sondern nur ein Parkplatz für das Fahrzeug. In einigen Fällen ist es dem Fahrzeug erlaubt, auch beladen ins Depot zu fahren. Ist das nicht möglich, so ist vor der Fahrt ins Depot noch eine IF zu besuchen. Hat ein Fahrzeug Bioabfälle geladen, so darf es nicht beladen im Depot stehen. Vor Wochenenden und Feiertagen sind die Fahrzeuge generell zu leeren, bevor sie im Depot abgestellt werden. Beladen ins Depot zu fahren, ist eine Neuerung und wird, soweit bekannt, in Österreich noch nicht praktiziert. Es wird erwartet, dass durch eine flexiblere Wahl, wann zu eine IF gefahren wird, sich kürzere Routen ergeben.

## 2.3 Formales Modell des MDPVRP-IF

Um allen Anforderungen der Tourenplanung für die Abfallsammlung in der Praxis gerecht zu werden, reicht das Modell des klassischem PVRP nicht aus. Das formale Modell des PVRP muss um einige Bedingungen erweitert werden. Und auch wenn die realen Probleminstanzen der periodischen Müllsammlung zu groß sind, um mittels linearer Programmierung gelöst zu werden, soll das Problem in diesem Abschnitt formal spezifiziert und damit geholfen werden, das Problem und vor allem die vielen Nebenbedingungen besser zu verstehen.

Fügt man zum PVRP die Möglichkeiten mehrere Depots zu haben und die Entleerung der Fahrzeuge in IFs vorzunehmen hinzu, erhält man das *Multi-Depot Periodic Vehicle Routing Problem with Intermediate Facilites (MDPVRP-IF)*. Im Folgenden wird ein formales Modell für das MDPVRP-IF vorgestellt, welches auf den Arbeiten [6, 9–12] aufbaut und um Nebenbedingungen, welche sich aus dem Kapitel 2.2 ergeben, erweitert wurde.

Das MDPVRP-IF wird als Multigraph $G = (V, A)$ modelliert. Depots ($D$), Zwischenlager ($F$) und Kunden ($N$) bilden die Knotenmenge $V = D \cup F \cup N$. Für Depots gilt, dass

sie in $V$ einmal als Ausgangspunkt einer Tour und einmal als Endpunkt einer Tour vorkommen $D = \{1, \ldots, d, d+1, \ldots, 2d\}$. Die Menge der Kanten $A$ wird beschrieben durch $A = \{(x, y)^{k,t} | x, y \in V, x \neq y, k \in K, t \in T\}$, wobei $k$ ein Fahrzeug aus der möglicherweise inhomogenen Fahrzeugflotte ($K$) beschreibt und $t$ für einen Tag im Planungszeitraum ($T$) steht. Für jede Kante existiert ein zugehöriger Wert $c_{ij}$, welcher die Distanz zwischen den Knoten $i$ und $j$ angibt. Distanz und Kosten werden in dieser Problemformulierung gleichgesetzt. Der Wert $c_{ij}$ kann so auch für die Kosten stehen, welche anfallen, wenn ein Fahrzeug vom Knoten $i$ zum Knoten $j$ fährt.

| Notation | Beschreibung |
|---|---|
| $a_{rt}$ | 1, wenn Tag $t$ im Besuchsmuster $r$ enthalten ist |
| $C_i$ | Menge der möglichen Besuchsmuster für den Kunden $i$ |
| $c_{ij}$ | Kosten der Kante ($i$, $j$) |
| $D$ | Menge der Depots |
| $D_k$ | Depot, in welchem Fahrzeug $k$ steht |
| $\epsilon$ | maximale Abweichung bei der Verhältniszuordnung |
| $F$ | Menge der IFs |
| $f_i$ | IF des Kunden $i$ bei der fixen Zuordnung |
| $f_{ikt}$ | zugewiesene IF für den Kunden $i$ und das Fahrzeug $k$ am Tag $t$ |
| $K$ | Menge der Fahrzeuge |
| $l_{ikt}$ | Ladung des Fahrzeugs $k$ am Tag $t$ nach dem Kunden $i$ |
| $N$ | Menge der Kunden |
| $Q_k$ | maximale Kapazität des Fahrzeugs $k$ |
| $q_i$ | die Menge Müll, die vom Kunden $i$ abgeholt wird |
| $s_i$ | Servicezeit beim Knoten $i$ |
| $T$ | Tage des Planungshorizonts |
| $V$ | Menge der Knoten ($D \cup F \cup N$) |
| $v_i$ | Prozentsatz für Verhältniszuordnung |
| $W_{kt}$ | maximale Tourlänge des Fahrzeugs $k$ am Tag $t$ |
| $w_{kt}$ | 1, wenn Fahrzeug $k$ am Tag $t$ unterwegs ist |
| $x_{ijkt}$ | 1, wenn Fahrzeug $k$ am Tag $t$ vom Kunden $i$ zum Kunden $j$ fährt |
| $y_{ir}$ | 1, wenn Besuchsmuster $r$ für den Kunden $i$ gewählt wurde |

**Tabelle 2.1:** Beschreibung der Variablen und Parameter des formalen Modells

### 2.3.1 Entscheidungsvariablen

Eine Lösung für das MDPVRP-IF wird über die zwei Entscheidungsvariablen $x_{ijkt}$ (2.1) und $y_{ir}$ (2.2) definiert. Aus diesen Variablen lassen sich die Besuchsmuster für die Kunden ablesen und für jeden Tag und jedes Fahrzeug ein Tourenplan herleiten.

$$x_{ijkt} = \begin{cases} 1 & \text{wenn Fahrzeug } k \text{ am Tag } t \text{ von Kunde } i \text{ zu Kunde } j \text{ fährt,} \\ 0 & \text{sonst.} \end{cases} \tag{2.1}$$

In der Variable $x_{ijkt}$ wird gespeichert, ob das Fahrzeug $k$ am Tag $t$ vom Kunden $i$ zum Kunden $j$ fährt. Die Reihenfolge ist dadurch nur implizit gegeben. Ein expliziter Plan kann leicht daraus ermittelt werden, da für jedes Fahrzeug das Depot, von dem aus es startet, bekannt ist.

$$y_{ir} = \begin{cases} 1 & \text{wenn Besuchsmuster } r \text{ aus } C_i \text{ für Kunden } i \text{ zutrifft,} \\ 0 & \text{sonst.} \end{cases} \tag{2.2}$$

Für Kunden existieren in fast allen Fällen mehrere Besuchsmuster. Besucht werden soll der Kunde aber nur nach genau einem Besuchsmuster. Welches Besuchsmuster für einen Kunden $i$ ausgewählt wurde, wird in der Variable $y_{ir}$ gespeichert.

### 2.3.2 Zielfunktion

Ziel ist es, die Lösung mit den geringsten Kosten zu finden. Der größte Kostenfaktor ist die Arbeitszeit, welche maßgeblich durch die zu fahrenden Strecken zustande kommt. Über die Zielfunktion (2.3) werden die Gesamtkosten ermittelt und zu minimieren versucht. Die Gesamtkosten ergeben sich durch die Summierung der Kosten ($c_{ij}$) aller gefahrenen Wegstrecken ($x_{ijkt}$).

$$\min \sum_{t \in T} \sum_{k \in K} \sum_{i \in V} \sum_{j \in V} x_{ijkt} \cdot c_{ij} \tag{2.3}$$

Folgende Nebenbindungen stellen sicher, dass trotz des Kostendrucks nicht alle Fahrzeuge in ihren Depots stehen bleiben, sondern der Müll der Kunden abgeholt wird.

### 2.3.3 Fahrzeuge

**Parameter:**

$D_k \in D \ldots$ Depot, in welchem Fahrzeug $k$ steht.
$Q_k \in \mathbb{N} \ldots$ maximale Kapazität des Fahrzeugs $k$.
$W_{kt} \in \mathbb{N} \ldots$ maximale Tourlänge des Fahrzeugs $k$ am Tag $t$.

$$w_{kt} = \begin{cases} 1 & \text{wenn Fahrzeug } k \text{ am Tag } t \text{ unterwegs ist } (W_{kt} > 0), \\ 0 & \text{sonst } (W_{kt} = 0). \end{cases}$$

**Variablen:**

$l_{ikt} \in \mathbb{N} \ldots$ Ladung des Fahrzeugs $k$ nach dem Besuch des Kunden $i$ am Tag $t$.

**Bedingungen:**

$$\sum_{j \in N \cup F} x_{gjkt} = 1 \qquad \forall k \in K, t \in T, g = D_k \tag{2.4}$$

$$\sum_{i \in N \cup F} x_{i(g+d)kt} = 1 \qquad \forall k \in K, t \in T, g = D_k \tag{2.5}$$

Jedes Fahrzeug fährt zu Beginn eines Arbeitstags aus dem Depot, in welchem es steht (2.4), und fährt am Ende des Arbeitstags auch wieder in dasselbe Depot zurück (2.5). Vom Depot-Knoten $i$ wird weggefahren und zum Depot-Knoten $i + d$ zurückgekehrt. Bei den Touren der Fahrzeuge handelt es sich um Pfade und nicht um Zyklen.

$$\sum_{i \in S} \sum_{j \in S} x_{ijkt} \leq |S| - 1 \qquad \forall k \in K, t \in T; S \subseteq V \backslash \{D\}; |S| \geq 2 \tag{2.6}$$

Bei der Gleichung (2.6) handelt es sich um die *Subtour-Eliminations-Nebenbedingung*. Diese ist notwendig, um Zyklen und nicht zusammenhänge Strecken in Touren zu verhindern.

$$l_{ikt} \leq Q_k \qquad \forall i \in V, k \in K, t \in T \tag{2.7}$$

Ein Fahrzeug darf zu keinem Zeitpunkt über seine maximale Kapazität hinaus beladen sein.

$$l_{ikt} = 0 \qquad \forall i \in F, k \in K, t \in T \tag{2.8}$$

Fährt ein Fahrzeug zu einem Zwischenlager, so hat das Fahrzeug danach nichts mehr geladen.

$$\sum_{i \in V} x_{ijkt} \cdot l_{ikt} + q_j = l_{jkt} \qquad \forall j \in D \cup N, k \in K, t \in T \tag{2.9}$$

Fährt ein Fahrzeug von einem Knoten $i$ zu einem Kunden oder ins Depot $j$, so ergibt sich ein neuer Ladestand aus dem alten Ladestand plus dem Müllvolumen des Kunden (bzw. plus null, falls zu einem Depot gefahren wird).

$$l_{(g+d)kt} = l_{gk(t+1)} \cdot w_{k(t+1)} \qquad \forall k \in K, t \in T, g = D_k \tag{2.10}$$

Fährt ein Fahrzeug beladen in das Depot, so hat es auch noch am nächsten Tag denselben Ladestand. Ist das Fahrzeug am nächsten Tag nicht im Einsatz, muss es am Vortag leer ins Depot fahren.

$$\sum_{j \in N \cup F} x_{gjkt} \cdot (c_{gj} + s_g) + \sum_{i \in N \cup F} \sum_{j \in N \cup F \cup D} x_{ijkt} \cdot (c_{ij} + s_j) \leq W_{kt} \quad \forall k \in K, t \in T, g = D_k \tag{2.11}$$

Für jedes Fahrzeug ist an jedem Tag eine maximale Tourlänge ($W_{kt}$) gegeben. Die Länge der gefahrenen Tour darf diesen Wert nicht überschreiten.

$$\sum_{i \in D \cup F \cup N} x_{ihkt} - \sum_{j \in D \cup F \cup N} x_{hjkt} = 0 \qquad \forall h \in F \cup N, k \in K, t \in T \tag{2.12}$$

Der Tag der Anreise zu einem Kunden bzw. einem Zwischenlager muss auch der Tag der Abreise sein.

$$\sum_{k \in K} \sum_{i \in D \cup F \cup N} \sum_{j \in N} x_{ijkt} \leq 1 \qquad \forall t \in T \tag{2.13}$$

Ein Kunde darf an jedem Tag nur maximal von einem Fahrzeug besucht werden. Es ist auch erlaubt, dass an bestimmten Tagen ein Kunde überhaupt nicht besucht wird. An welchen Tagen ein Kunde besucht wird, hängt vom gewählten Besuchsmuster ab (siehe Kapitel 2.3.6).

### 2.3.4 Depots

**Parameter:**

$s_i \in \mathbb{N} \ldots$ Servicezeit eines Fahrzeugs im Depot $i$.
$q_i = 0 \ldots$ in einem Depot wird weder Müll abgeholt noch das Fahrzeug entleert.

### 2.3.5 Intermediate Facilities

**Parameter:**

$s_i \in \mathbb{N} \ldots$ Servicezeit zum Entladen eines Fahrzeugs beim Zwischenlager $i$.

Das Müllaufkommen kann auf fünf verschiedene Arten auf die Zwischenlager verteilt werden.

#### 2.3.5.1 Fixe Zuordnung für alle Kunden

**Parameter:**

$f_i \in D \ldots$ jedem Kunden $i$ ist ein Zwischenlager zugeteilt.

**Bedingungen:**

$$\sum_{i \in N} \sum_{j \in N} (f_i - f_j) \cdot x_{ijkt} = 0 \qquad \forall k \in K, t \in T \tag{2.14}$$

Wenn Kunde $j$ nach dem Kunden $i$ besucht wird, so muss Kunde $j$ dasselbe Zwischenlager haben wie Kunde $i$.

$$\sum_{i \in N} \sum_{j \in N} x_{i(g+d)kt} \cdot x_{gjk(t+1)} \cdot f_i - f_j) = 0 \qquad \forall k \in K, t \in T, g = D_k \tag{2.15}$$

Auch wenn das Fahrzeug nach einem Kunden ins Depot fährt, muss am nächsten Tag der erste Kunde noch dasselbe Zwischenlager wie der letzte Kunde des Vortags haben.

$$x_{ihkt} \cdot (f_i - h) = 0 \qquad \forall i \in N, h \in F, k \in K, t \in T \tag{2.16}$$

$$\sum_{i \in N} x_{i(g+d)kt} \cdot x_{ghk(t+1)} \cdot (f_i - h) = 0 \qquad \forall h \in F, k \in K, t \in T, g = D_k \tag{2.17}$$

Fährt das Fahrzeug $k$ zum Zwischenlager $h$, so muss der zuletzt besuchte Kunde dieses Zwischenlager $h$ auch zugeteilt bekommen haben (2.16). Dies muss auch gelten, wenn dazwischen das Fahrzeug ins Depot gefahren ist (2.17).

#### 2.3.5.2 Fixe Zuordnung nur für einige Kunden

**Parameter:**

$$f_i = \begin{cases} f \in F & \text{wenn dem Kunden } i \text{ ein Zwischenlager } f \text{ zugeteilt ist,} \\ 0 & \text{sonst.} \end{cases}$$

**Variablen:**

$f_{ikt} \in F \cup 0 \ldots$ Zwischenlager, zu welchem der Müll des Kunden $i$ am Tag $t$ von Fahrzeug $k$ gebracht werden soll.

**Bedingungen:**

$$f_{ikt} = 0 \qquad \forall i \in F, \forall k \in K, \forall t \in T \tag{2.18}$$

Nach einem Zwischenlager $i$ kann ein beliebiges anderes Zwischenlager besucht werden, sofern kein Kunde auf der Tour ein bestimmtes Zwischenlager vorschreibt.

$$f_{jkt} = \sum_{i \in V} x_{ijkt} \cdot f_{ikt} \qquad \forall j \in N \cup D, \forall k \in K, \forall t \in T \tag{2.19}$$

Zu welchem Zwischenlager der Abfall von Kunden $j$ am Tag $t$ gebracht werden soll, hängt davon ab, zu welchem Zwischenlager der Abfall des Kunden gebracht werden soll, der vor dem Kunden $j$ besucht wurde.

$$f_{(g+d)kt} = f_{gk(t+1)} \qquad \forall k \in K, \forall t \in T, g \in D \tag{2.20}$$

Fährt ein Fahrzeug beladen in das Depot, so muss es am nächsten Tag bei der nächsten Entleerung das Zwischenlager verwenden, welches am Vortag vorgegeben wurde (2.20). Wenn das Fahrzeug beladen in das Depot fährt, kann nicht der letzte Tag sein, und wenn das Fahrzeug leer ins Depot fährt, wurde direkt davor eine IF besucht und es gilt $f_{gk(t+1)} = 0$.

$$\sum_{i \in D \cup N} x_{ijkt} \cdot f_j \cdot f_{ikt} \cdot f_{jkt} \cdot (f_{ikt} - f_{jkt}) = 0 \qquad \forall j \in D \cup N, \forall k \in K, \forall t \in T \tag{2.21}$$

Für alle Kunden muss gewährleistet sein, dass der Müll, wenn ein Zwischenlager vorgegeben ist, auch tatsächlich zu diesem Zwischenlager gebracht wird. Für andere Kunden, die noch auf dem Weg zum Zwischenlager liegen, muss gelten, dass ihnen entweder kein Zwischenlager zugewiesen ist oder es sich um dasselbe Zwischenlager handelt.

#### 2.3.5.3 Verhältniszuordnung

**Parameter:**

$v_i \in \mathbb{R}^+ < 1 \ldots$ Prozentsatz des Gesamtmüllvolumens, welcher zum Zwischenlager $i$ gebracht werden soll.
$\epsilon \in \mathbb{R}^+ \ldots$ Betrag, um welchen die gebrachte Müllmenge abweichen darf.

**Bedingungen:**

$$\left| \sum_{t \in T} \sum_{k \in K} \sum_{i \in D \cup N} x_{ifkt} \cdot l_{ikt} - v_f \cdot \left( \sum_{i' \in N} q_{i'} \right) \right| \leq \epsilon \qquad \forall f \in F \tag{2.22}$$

Über alle Tage und alle Fahrzeuge muss die zum Zwischenlager $f$ gebrachte Müllmenge dem Prozentsatz $v_f$ des Gesamtmüllvolumens entsprechen. Es ist nur eine Abweichung von maximal $\epsilon$ erlaubt.

#### 2.3.5.4 Gemischte Zuordnung

Bei der gemischten Zuordnung sind für einige Kunden die Zwischenlager fix vorgegeben. Der Abfall dieser Kunden muss zum angegebenen Zwischenlager gebracht werden. Es kann dabei auch Müll von anderen Kunden mitgenommen werden, welche kein Zwischenlager vorgegeben haben. Über den Planungszeitraum muss die Verteilung des Mülls auf die Zwischenlager nach einem gewissem Schlüssel geschehen. Es handelt sich also um eine Kombination aus der fixen Zuordnung für einige Kunden und der Verhältniszuordnung. Um diesen Modus abzudecken, sind die Nebenbedingungen aus Kapitel 2.3.5.2 und 2.3.5.3 zu verwenden.

#### 2.3.5.5 Freie Zuordnung

Bei der freien Zuordnung sind die Zwischenlager frei wählbar, daher sind keine zusätzlichen Nebenbedingungen notwendig. Die Zielfunktion stellt sicher, dass das Zwischenlager immer so gewählt wird, dass der Umweg so gering wie möglich ist und die Zusatzkosten gering gehalten werden.

### 2.3.6 Kunden

**Parameter:**

$q_i \in \mathbb{N} \ldots$ die Menge Müll, die vom Kunden $i$ abgeholt wird.
$s_i \in \mathbb{N} \ldots$ Servicezeit beim Kunden $i$.
$C_i \ldots$ die Menge der möglichen Besuchsmuster für den Kunden $i$.

$$a_{rt} = \begin{cases} 1 & \text{wenn der Tag } t \text{ in dem Besuchsmuster } r \text{ enthalten ist,} \\ 0 & \text{sonst.} \end{cases}$$

**Bedingungen:**

$$\sum_{r \in C_i} y_{ir} = 1 \qquad \forall i \in N \tag{2.23}$$

Für jeden Kunden muss genau ein Besuchsmuster ausgewählt werden. Es darf keinen Kunden geben, welchem kein Besuchsmuster oder mehrere Besuchsmuster zugeordnet wurden.

$$\sum_{k \in K} \sum_{j \in V} x_{ijkt} - \sum_{r \in C_i} a_{rt} \cdot y_{ir} = 0 \qquad \forall i \in N, t \in T \tag{2.24}$$

Ein Kunde darf nur an den Tagen besucht werden, welche das Besuchsmuster vorschreibt, und er muss an diesen Tagen auch besucht werden.

Zwecks Übersichtlichkeit und Verständlichkeit wurde darauf verzichtet, die Zeitfenster, welche nicht in allen Probleminstanzen vorhanden sind und dann auch nur für wenige Kunden existieren, in das formale Modell aufzunehmen.

KAPITEL 3

# Verwandte Arbeiten

Zum Thema PVRP wurden bereits einige wissenschaftliche Publikationen veröffentlicht und viele davon beschäftigten sich auch mit der Müllentsorgung. Dieses Kapitel soll einige wichtige Arbeiten und deren Lösungsansätze präsentieren. Für einen ausführlicheren Überblick über den aktuellen Forschungsstand sei auf die Arbeit „The period vehicle routing problem and its extensions“ [6] von P.M. Francis et al. verwiesen.

## 3.1 Assignment Routing Problem

R. Russel und W. Igo haben in ihrem Artikel „An Assignment Routing Problem“ [7] 1979 zum ersten Mal das *Assignment Routing Problem (ARP)* definiert und einen Lösungsansatz präsentiert. Sie betrachteten dazu das Problem der industriellen Abfallentsorgung mit vier Müllkompressor-Fahrzeugen. Dabei übernahmen sie die Besuchsmuster vom Auftraggeber und optimierten nur die Tagestouren. Das Ziel war, das Einsparungspotential zu ermitteln, wenn auch die Besuchsmuster frei gewählt werden können. Die Probleminstanz umfasste 490 Kunden, welche zwischen ein und sechsmal im Zeitraum von einer Woche besucht werden sollten. In Summe gab es 776 Punkte, zu denen die Fahrzeuge fahren sollten.

Das Problem wurde von den Autoren als ARP bezeichnet, da die Auswahl der Besuchsmuster für die Kunden im Vordergrund stand. Da es sich um eine Verallgemeinerung des VRP handelt und dieses bereits $NP$-vollständig ist, gilt dies auch für das ARP. Ein *Mixed-Integer-Modell* wurde zwar formuliert, aber der exakte Lösungsansatz nicht weiter verfolgt, sondern es wurden drei Heuristiken entwickelt.

Die erste Heuristik generiert und verbessert Lösungen durch Clustern über mehrere Iterationen. In der ersten Iteration werden die Kunden, welche jeden Tag besucht werden müssen, als Zentrum eines Clusters festgelegt. Die restlichen Kunden werden diesen Zentren zugeordnet. Die Zuordnung erfolgt nur, wenn ein Besuchsmuster kürzere Wege für alle Tage liefert als alle anderen möglichen Besuchsmuster. Dadurch entsteht in der ersten Iteration eine vollständige Lösung. In den weiteren Iterationen (ca. vier) wird für jeden Kunden untersucht, ob in der voll-

ständigen Lösung nicht doch ein anderes Besuchsmuster besser ist. Bei allen Zuordnungen werden die Nebenbedingungen (Kapazitätsbeschränkungen der Fahrzeuge etc.) eingehalten. Diese Heuristik ist schnell und liefert gültige Lösungen, reduziert jedoch die zu fahrenden Kilometer nicht sehr. Daher wird sie nur als Konstruktionsheuristik für die zweite Heuristik verwendet.

Bei der zweiten Heuristik handelt es sich um eine Modifikation der MTOUR-Heuristik, welche wieder eine Erweiterung der *Lin-und-Kernighan-Heuristik* für das VRP ist. Auch wenn diese Heuristik gute Ergebnisse liefert, ist sie wegen der längeren Laufzeit nicht für große Probleme geeignet.

Als dritte Heuristik wurde der Savings-Ansatz von Clarke and Wright modifiziert, sodass alle Nebenbedingungen erfüllt bleiben. Als initiale Besuchsmusterzuweisung wurde wieder die erste Heuristik verwendet. Um die große Probleminstanz lösen zu können, wurde die Anzahl der Nachbarn, für die das Einsparpotential berechnet wurde, auf die 40, 60 und 70 nächsten Nachbarn beschränkt. Für diese Parameter konnten Einsparungen von 5,3%, 6,8% und 7,5% erzielt werden. Die Laufzeit betrug um die 20 Minuten auf einem IBM 370/158. Der MTOUR-Ansatz schaffte 4,8% Streckenreduktion in 6 Minuten.

Das Resümee der Autoren ist, dass das Einsparungspotential stark von den Daten abhängt. Vor allem die Flexibilität bei den Besuchsmustern ist entscheidend für die Optimierung. Ist diese aber gegeben, können durch die frei(ere) Wahl der Besuchsmuster Verbesserungen erzielt werden.

## 3.2 Tabu Search Heuristic for Periodic and Multi-Depot Vehicle Routing Problems

In [13] stellen J.-F. Cordeau, M. Gendreau und G. Laporte eine *Tabu-Suche (TS)* für das PVRP, das PTSP und das *Multi-Depot Vehicle Routing Problem (MDVRP)* vor. Die Autoren zeigen, dass sowohl das PTSP als auch das MDVRP als PVRP modelliert werden können. In den Tests hat die vorgestellte TS gezeigt, dass sie für alle drei Probleme geeignet ist und für fast alle in der Literatur verwendeten Testinstanzen bessere Lösungen findet. Im Vergleich zu anderen Verfahren benötigt die TS aber mehr Zeit und laut den Autoren ist es wahrscheinlich, dass einige der anderen Verfahren mit mehr Zeit ebenfalls bessere Lösungen finden könnten.

Den Autoren ist es damit gelungen, eine TS zu entwickeln, welche mit wenigen Parametern auskommt und trotzdem sehr gute Ergebnisse in vernünftiger Zeit liefert.

## 3.3 Periodic Vehicle Routing Problem with Intermediate Facilities

Immer, wenn Güter ausgeliefert werden sollen und diese Güter in einem Warenlager sind und nicht im Depot, müssen die Fahrzeuge vorher zum Warenlager fahren, bevor mit der Auslieferung begonnen werden kann. Hat ein Fahrzeug die geladenen Güter ausgeliefert, kann es wieder zu einem Warenlager fahren und neuerlich beladen werden. Diese Problemstellung lässt sich nicht mit dem klassischem VRP abbilden. Es kann aber um Warenlager erweitert werden und funktioniert sowohl für die Auslieferung als auch für das Einsammeln von Gütern. In der Literatur wird dies als *Vehicle Routing Problem with Intermediate Facilities (VRP-IF)* bezeichnet.

E. Angelelli und M. G. Speranza formulierten das *Periodic Vehicle Routing Problem with Intermediate Facilities (PVRP-IF)* und erweiterten die TS von J.-F. Cordeau et al. (siehe Kapitel 3.2) für das PVRP-IF. Um das PVRP-IF lösen zu können, haben die Autoren einerseits eine andere Heuristik verwendet, um eine Startlösung zu generieren, und bei der TS zusätzliche Nachbarschaften definiert. In einigen Fällen konnte diese TS bei PVRP Instanzen geringfügig bessere Lösungen finden als die TS von J.-F. Cordeau et al. Mangels Vergleichswerten für Probleme mit IFs kann über die Qualität der Lösungen für das PVRP-IF keine Aussage getroffen werden.

## 3.4 Waste Collection Vehicle Routing Problem with Time Windows

Für ein großes Entsorgungsunternehmen in Nordamerika entwickelten B.-I. Kim, S. Kim und S. Sahoo ein Verfahren [14], das die Entleerung von Müllcontainern bei Gewerbebetrieben plant und dabei Zeitfenster berücksichtigt. Im Konkreten handelt es sich um ein *Vehicle Routing Problem with Time Windows and Intermediate Facilities (VRPTW-IF)* mit mehreren Zielgrößen. Die Fahrzeuge starten von einem Depot und entleeren Abfallcontainer bei den Kunden. Hat ein Kunde ein Zeitfenster für die Entleerung angegeben, so ist eine Entleerung nur während des Zeitfensters gestattet. Hat ein Fahrzeug seine maximale Ladekapazität erreicht, so ist zu einer IF zu fahren. Die Mittagspause des Personals wird für jedes Fahrzeug separat festgelegt, hat zwischen 11 und 12 Uhr zu beginnen und dauert eine Stunde.

Ziel ist es, alle Kunden mit so wenig Fahrzeugen wie möglich zu besuchen und die Einsatzzeiten des Personals gering zu halten. Einen sehr hohen Stellenwert haben auch die Kompaktheit der Touren und die ausgeglichenen Einsatzzeiten zwischen den einzelnen Fahrzeugen.

Zur Lösungsfindung werden die Kunden in Gruppen zusammengefasst, und mittels Einfügeheuristik wird für jede Gruppe eine Tour festgelegt. Im nächsten Schritt wird durch eine *Simulated Annealing-Heuristik (SA)* versucht, die Kompaktheit der Touren durch das Austauschen von Kunden zwischen zwei Touren zu verbessern. Dann erst wird für jede Tour einzeln die Einsatzzeit optimiert.

Das Verfahren ist seit 2004 im Einsatz und konnte bereits im ersten Jahr mehrere Millionen Dollar einsparen. Nun soll es auch in anderen Bereichen eingesetzt werden.

## 3.5 Variable Neighborhood Search for Periodic Vehicle Routing Problems

V. C. Hemmelmayr, K. F. Doerner und R. F. Hartl stellen in [3] als Erste ein *Variable Neighborhood Search-Verfahren (VNS)* für das PVRP vor. Ausgehend von einer Startlösung, die von einer *Savings-Heuristik* erstellt wird, sucht die VNS-Heuristik nach besseren Lösungen. Dabei werden die Besuchsmuster der Kunden verändert und die Touren der Fahrzeuge neu zusammengestellt. Bei diesen Veränderungen können immer wieder ungültige Lösungen entstehen. Durch Strafkosten für ungültige Lösungen und die Kombination der VNS mit einem SA, werden trotzdem rasch gültige Lösungen gefunden und der Lösungsraum gut durchsucht.

Getestet wurde das Verfahren auf bereits bekannten Datensets, wie zum Beispiel jenem aus der Arbeit von J.-F. Cordeau et al. [13]. Bei einigen Testinstanzen handelt es sich um dege-

nerierte PVRPs, d.h. VRPs (nur für einen Tag) oder PTSPs (nur ein Fahrzeug). Trotz dieser Extremfälle erzielte das Verfahren bei 37 der 42 Testinstanzen gleich gute oder bessere Werte als alle vergleichbaren Verfahren. Achtet man auch auf vergleichbare Laufzeiten, so konnten noch immerhin für 16 der 42 Instanzen neue Bestwerte gefunden und für 7 Instanzen zumindest die früheren Bestwerte erreicht werden.

KAPITEL 4

# Lösungsansätze

Im vorigen Kapitel wurden Arbeiten vorgestellt, welche ähnliche Probleme analysiert und gelöst haben. Aufgabe dieses Kapitels soll es nun sein, zu erklären, wie das PVRP der Müllentsorgung gelöst wurde, welche Verfahren verwendet wurden und warum genau diese. Es soll auch auf die Vor- und Nachteile dieser Verfahren eingegangen werden.

Die zu lösenden Probleminstanzen haben zwischen 50 und 500 Kunden und eine Fahrzeugflotte bestehend aus mehreren Fahrzeugen mit möglicherweise unterschiedlichen Depots, verschiedenen maximalen Ladekapazitäten und teilweise abweichende tägliche Einsatzzeiten. Bei Problemen dieser Größe und Komplexität sind exakte Verfahren nicht praktikabel. Zu lange würde das Finden der optimalen Lösung dauern. Es kommen daher Approximationsalgorithmen zum Einsatz. Dabei wird akzeptiert, dass dadurch keine Garantie über die Güte der Lösungsqualität abgegeben werden kann. Dafür kann in kürzerer Zeit mit Lösungen gerechnet werden.

Aus den vorgestellten Verfahren im Kapitel 3 eignet sich besonders der Ansatz von V. Hemmelmayr et al. in der Arbeit „A Variable Neighborhood Search Heuristic for Periodic Routing Problems“ [3]. Die Wahl auf das VNS fiel aus folgenden Gründen:

- **Einfachheit:** Das VNS ist einfach und modular im Aufbau und daher auch leicht zu verstehen, zu erweitern und zu implementieren.
- **Robustheit:** Die Lösungsqualität selbst von unbekannten Probleminstanzen ist gut, auch wenn diese Extremfälle des PVRP darstellen (PTSP und VRP).
- **Generalität:** Das VNS beinhaltet kein problemspezifisches Wissen. Das problemspezifische Wissen ist in der Lösungsrepräsentation enthalten und getrennt vom VNS. Eine Implementierung ist daher einfach und eine Erweiterung oder Anpassung des Problems zieht nicht gleichzeitig auch eine notwendige Änderung des VNS nach sich.
- **Flexibilität:** Es gibt viele Versionen des VNS und allgemeine Erweiterungen, welche für jede Version passen. Es ist daher möglich, auf spezielle Problemanforderungen einzugehen.

- **Lösungsqualität:** Das VNS hat bei einem ähnlichen Problem bereits sehr gute Lösungen geliefert und auch bei vielen anderen Problemen eine gute Performanz gezeigt.

- **Laufzeit:** Das Verhältnis von Lösungsqualität zu Laufzeit ist leicht über Parameter zu steuern und liefert auch bei kurzer Laufzeit gute Ergebnisse.

Das VNS von V. Hemmelmayr et al. löst das klassische PVRP. Dabei werden keine IFs und Zuordnungsmodi unterstützt. Es gibt nur ein Depot und die Fahrzeuge können nicht beladen ins Depot fahren. Das Problem der Müllentsorgung erfordert es, dass auch diese Faktoren berücksichtigt werden. Folglich ist es notwendig, das VNS zu erweitern.

Im restlichen Kapitel werden die Heuristiken vorgestellt, mit denen das PVRP der Müllentsorgung gelöst werden kann. Dabei wird jedes Verfahren allgemein erklärt, bevor auf die notwendigen Erweiterungen eingegangen wird. Kapitel 4.1 stellt die Konstruktionsheuristik von G. Clarke und J. W. Wright vor. Es folgt im Kapitel 4.2 eine ausführliche Erklärung des Verbesserungsverfahrens VNS von N. Mladenović und P. Hansen.

## 4.1 Savings-Heuristik

Die Savings-Heuristik ist ein Konstruktionsverfahren aus dem Jahr 1964, welches für die Tourenplanung verwendet wird. Sie wird nach ihren Entwicklern, Clarke und Wright, auch als Clarke-and-Wright-Heuristik bezeichnet [1, 15]. Wie für heuristische Verfahren charakteristisch, wird zwar eine optimale Lösung angestrebt, diese kann aber nicht garantiert werden. Die Funktionsweise soll anhand der Auslieferung von Waren aus einem Depot zu Kunden demonstriert werden.

Die Idee der Savings-Heuristik ist es, anfangs jeden Kunden einzeln zu besuchen und dann wieder zum Depot zurückkehren $(d, j, d)$. Man spricht dabei von Pendeltouren. Eine Kombination von zwei Pendeltouren zu $j$ und $k$ führt zu Einsparungen (engl. savings), wenn die Distanz zwischen $j$ und $k$ kürzer ist als die Distanzen von und zum Depot (siehe Abbildung 4.1). Die zwei Pendeltouren können durch eine Tour $(d, j, k, d)$ ersetzt werden. Die Ersparnis berechnet sich folgendermaßen:

$$s_{i,j} = c_{i,d} + c_{d,j} - c_{i,j} \tag{4.1}$$

Doch auch längere Touren, bestehend aus mehreren Kunden, die besucht werden, lassen sich auf diese Weise zusammenführen, wenn die Fahrt vom Depot der Tour $l$ und die Fahrt zum Depot der Tour $m$ länger sind als eine Kombination beider Touren (siehe Abbildung 4.2). Von allen möglichen Kombinationen von Touren wird immer diejenige gewählt, welche die größten Einsparungen erzielt und keine Überschreitung der maximalen Einsatzzeit verursacht.

Bei jedem Kombinationsschritt verringert sich die Anzahl der Touren um eins. Der Algorithmus terminiert, wenn jedem Fahrzeug eine Tour zugewiesen werden kann und keine Tour übrig bleibt.

Die Savings-Heuristik ist für das eintägige VRP entwickelt worden und bedarf daher einiger Anpassungen, damit sie auch für das hier zu lösende PVRP mit all seinen Nebenbedingungen

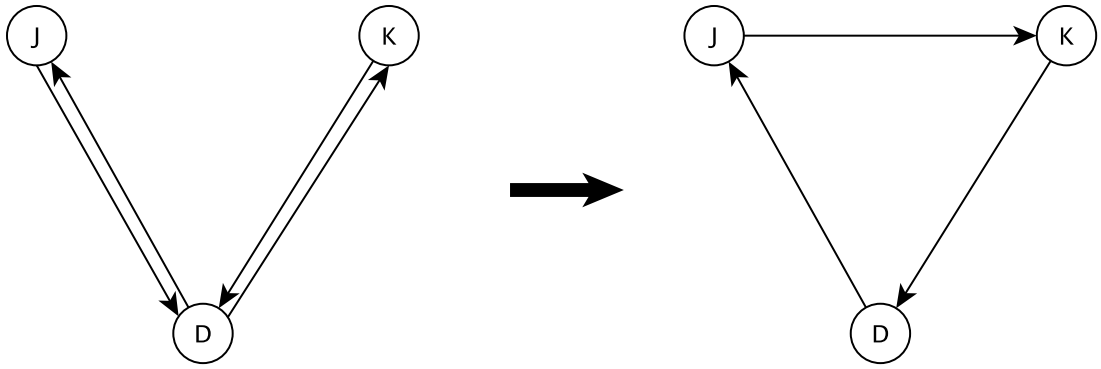



**Abbildung 4.1:** Prinzip der Savings-Heuristik (Abbildung nach [1])

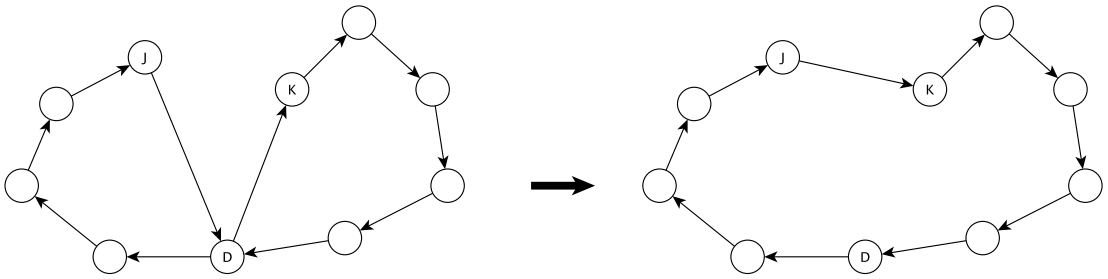



**Abbildung 4.2:** Prinzip der Savings-Heuristik bei längeren Touren (Abbildung nach [1])

funktioniert. Da das PVRP immer mehrere Tage umfasst, muss die Savings-Heuristik für jeden Tag des Planungszeitraums getrennt angewendet werden.

Auf die unterschiedlichen Einsatzzeiten der Fahrzeuge kann nur teilweise eingegangen werden. Es wird immer die längste Einsatzzeit des Tages gewählt und für alle Touren festgelegt. Wenn die Kombination von zwei Touren die maximal erlaubte Einsatzzeit überschreitet, dann wird die Länge der Tour von der Ersparnis abgezogen. Nur wenn es sich weiterhin um die beste Kombination handelt, wird diese Überschreitung zugelassen. Auf die Ladungskapazität und die weiteren Einschränkungen kann dabei keine Rücksicht genommen werden. Der genaue Ablauf bei der Savings-Heuristik für das PVRP ist in Algorithmus 4.1 dargestellt.

Die Lösungen, die dabei entstehen, sind mit großer Wahrscheinlichkeit nicht gültig und verletzen einige oder sehr viele der Nebenbedingungen. Eine gültige Startlösung zu produzieren, ist mit sehr viel Mehraufwand verbunden und auf Grund des unsicheren Mehrwerts nicht gewollt. Aufgabe der Savings-Heuristik ist es, nur eine zulässige Startlösung zu produzieren, mit der eine Verbesserungsheuristik arbeiten kann.

**Algorithmus 4.1:** Savings-Heuristik für das PVRP

**Eingabe** : ein Set von Fahrzeugen und ihre Einsatzzeiten für jeden Tag; eine Menge von Kunden und ihre Besuchsmuster; ein Depot $d$

**Ausgabe** : eine zulässige Lösung, in der alle Kunden entsprechend ihrer Besuchsmuster Fahrzeugen zugeordnet sind

```
1  für alle k ∈ Kunden tue
2      wähle ein zufälliges Besuchsmuster für k aus
3  für alle t ∈ Tagen tue
4      z_max ← maximale Einsatzzeit für t
5      für alle k ∈ Kunden tue
6          pt ← erstelle eine Pendeltour
7          füge pt zu den Touren hinzu
8      wiederhole
9          für alle x ∈ Touren tue
10             für alle y ∈ Touren tue
11                 s_{x,y} ← berechne Einsparung
12                 wenn Einsatzzeit von x + y > z_max dann
13                     s_{x,y} ← s_{x,y} − Einsatzzeit von x + y
14         kombiniere Tour mit höchstem Einsparungspotential
15     bis Anzahl der Touren ≤ Anzahl der Fahrzeuge
16     weise jedem Fahrzeug eine Tour zu
17 s ← erstelle Lösung aus Fahrzeugen und Touren
18 zurück s
```

## 4.2 Variable Neighborhood Search

1997 wurde das VNS Rahmenwerk zum ersten Mal von N. Mladenovic und P. Hansen vorgestellt [16]. Die Autoren beschrieben die große Flexibilität der vorgestellten Methode. Mittlerweile gibt es unzählige Varianten des VNS mit unterschiedlichen Erweiterungen. Zur besseren Einteilung der verschiedenen Varianten wird das ursprüngliche VNS von den Autoren als *Basic Variable Neighborhood Search (BVNS)* bezeichnet. Diese Arbeit hält sich im Folgenden auch an diese Konvention.

Ein essentieller Grundbaustein für das VNS sind die Nachbarschaften. Als Nachbarschaft einer Ausgangslösung ($N(x)$) versteht man die Menge der Lösungen, welche durch eine bestimmte Veränderung der Ausgangslösung entstehen. Die Veränderung kann aus einem Schritt oder mehreren Schritten bestehen.

Dazu das folgende Beispiel: Angenommen, die Ausgangslösung $x$ ist repräsentiert durch

einen binären Vektor der Länge 5: $(v_1, v_2, v_3, v_4, v_5)$. In der Ausgangslösung sind alle Variablenwerte null. Eine kleine Nachbarschaft $N_1$ wäre, wenn einer der Variablenwerte geändert wird.

$$N_1(x) = \{(1,0,0,0,0), (0,1,0,0,0), (0,0,1,0,0), \dots\}. \tag{4.2}$$

Eine größere Nachbarschaft $N_2$ wäre, wenn immer genau drei Variablenwerte geändert werden.

$$N_2(x) = \{(1,1,1,0,0), (1,1,0,1,0), (1,1,0,0,1), (1,0,1,1,0), \dots\}. \tag{4.3}$$

$N_2$ ist größer als $N_1$, da die Anzahl der Lösungen in $N_2(x)$ größer ist als in $N_1(x)$. Keine der beiden Nachbarschaften kann alle möglichen Nachbarn von $x$ erzeugen.

Der Vorteil von kleinen Nachbarschaften ist, dass sie schneller durchsucht werden können. Große Nachbarschaften decken dafür einen größeren Bereich ab, womöglich sogar den gesamten Suchraum. Lokalen Optima kann so entkommen werden.

Die folgenden drei Beobachtungen über Nachbarschaften verwendeten Mladenović und Hansen 1997 [2, 16] als Grundlage für das VNS:

1. Ein lokales Minimum einer Nachbarschaft ist nicht unbedingt ein lokales Minimum einer anderen Nachbarschaft.

2. Ein globales Minimum ist ein lokales Minimum in allen möglichen Nachbarschaften.

3. Bei vielen Problemen liegen lokale Minima einer oder mehrerer Nachbarschaften eng beisammen.

Die Idee von VNS ist nun, mehrere Nachbarschaften bei der Optimierung einzusetzen. Die Nachbarschaften werden systematisch gewechselt. Ist das Minimum einer Nachbarschaft gefunden, wird zur nächsten Nachbarschaft gewechselt. Es besteht die Möglichkeit, dass diese Nachbarschaft ein noch besseres Minimum findet (Folgerung aus der ersten Beobachtung). Sollte keine bessere Lösung gefunden werden, so ist bezüglich der gewählten Nachbarschaften die Optimallösung gefunden (Folgerung aus der zweiten Beobachtung). Lokale Optima enthalten bei vielen Problemen Teile der Optimallösung. Die Schwierigkeit liegt darin, herauszufinden, welche Teile der Lösung auch in der Optimallösung vorkommen. Nachbarschaften, welche diesen Umstand ausnutzen können, finden schneller bessere Lösungen (Folgerung aus der dritten Beobachtung).

Nach der Auswahl einer Lösung aus einer Nachbarschaft wird diese evaluiert. Das Ergebnis der Bewertung wird herangezogen, um zu entscheiden, ob diese Lösung akzeptiert wird, d.h. als neuer Ausgangspunkt der Suche gewählt wird. Das Akzeptanzkriterium ist bei den meisten VNS Varianten, dass nur verbessernde Lösungen angenommen werden.

Es folgen Beschreibungen von Variable Neighborhood Descent (VND), Reduced Variable Neighborhood Search (RVNS), Basic Variable Neighborhood Search (BVNS), Skewed Variable Neighborhood Search (SVNS) und die Beschreibung eines adaptierten VNS, mit welchem das PVRP gelöst wurde.

### 4.2.1 Variable Neighborhood Descent (VND)

Bei *Variable Neighborhood Descent (VND)* werden die Nachbarschaften $N$ deterministisch gewechselt. Dazu werden die Nachbarschaften von 1 bis $k_{max}$ durchnummeriert.

Begonnen wird mit der Startlösung und der ersten Nachbarschaft $N_1$. Es wird immer die komplette Nachbarschaft durchsucht. Wird eine bessere Lösung gefunden, so wird diese Lösung übernommen und in die erste Nachbarschaft gewechselt. Ist die beste Lösung einer Nachbarschaft $N_x$ schlechter als die derzeit beste Lösung, so wird in die nächste Nachbarschaft $N_{x+1}$ gewechselt. Ist auch die letzte Nachbarschaft $N_{k_{max}}$ durchsucht, so terminiert der VND-Algorithmus. Der Pseudocode für VND ist in Algorithmus 4.2 nachzulesen.

**Algorithmus 4.2:** Variable Neighborhood Descent (VND) [5]

**Eingabe** : eine Startlösung $S$ und ein Set von Nachbarschaften $N_k$, wobei $k = 1, \ldots, k_{max}$

**Ausgabe** : eine verbesserte Ausgangslösung, wenn Verbesserungen gefunden werden konnten, sonst die unveränderte Ausgangslösung

```
1  k ← 1
2  wiederhole
3      finde S' ∈ N_k mit f(S') < f(y), ∀y ∈ N_k
4      wenn f(S') < f(S) dann
5          S ← S'
6          k ← 1
7      sonst
8          k ← k + 1
9  bis k = k_max
10 zurück S
```

Das Resultat ist eine Lösung, die in allen Nachbarschaften von $N$ ein lokales Minimum darstellt. Sind die Nachbarschaften gut gewählt, so kann dies sogar die Optimallösung sein. Das Resultat und die Laufzeit hängen stark von der Startlösung ab. Im schlechtesten Fall müssen alle Lösungen mindestens einmal (wahrscheinlicher ist mehrmals) berechnet werden.

Bei vielen $NP$-schweren Problemen mit größeren Probleminstanzen benötigt die VND zu lange, um gute Lösungen zu finden. In vielen dieser Fälle hat sich das VNS bewährt.

### 4.2.2 Reduced Variable Neighborhood Search (RVNS)

Während bei dem VND die komplette Nachbarschaft durchsucht wird, wird bei dem *Reduced Variable Neighborhood Search (RVNS)* nur eine einzige Lösung aus der Nachbarschaft ausgesucht und bewertet (siehe Algorithmus 4.3). Die Auswahl erfolgt durch Zufall. Bessere Lösungen werden übernommen, schlechtere verworfen.

Durch die zufällige Auswahl von neuen Lösungen ist dem RVNS zu keinem Zeitpunkt bekannt, ob schon alle Lösungen zumindest einmal evaluiert wurden. Das Verfahren würde auf der Suche nach noch besseren Lösungen nie terminieren. Es ist daher notwendig, eine oder mehrere Bedingungen zu wählen, die zum Abbruch des Algorithmus führen. Mögliche Abbruchkriterien sind:

- Laufzeit: Es wird eine gewisse Zeit festgelegt, die der Algorithmus maximal laufen darf.
- Iterationen: Wird die maximale Anzahl von Iteration erreicht, wird die Berechnung beendet, egal, wie viel Zeit verstrichen ist.
- Iterationen ohne Verbesserung: Wird für eine bestimmte Anzahl von Iterationen keine bessere Lösung gefunden, terminiert der Algorithmus.
- Lösungsqualität: Hat die Lösung eine gewisse Güte erreicht, ist zu erwarten, dass es keine besseren Lösungen mehr gibt oder die noch möglichen Verbesserungen den Mehraufwand nicht rechtfertigen.

Einige Kombinationen aus den aufgelisteten Abbruchkriterien sind in bestimmen Situationen von Interesse.

**Algorithmus 4.3:** Reduced Variable Neighborhood Search (RVNS)

**Eingabe** : eine Startlösung $S$, ein Set von Nachbarschaften $N_k$, wobei $k = 1, \ldots, k_{max}$, und ein Abbruchkriterium

**Ausgabe** : eine verbesserte Ausgangslösung, wenn Verbesserungen gefunden werden konnten, sonst die unveränderte Ausgangslösung

```
1  wiederhole
2      k ← 1
3      wiederhole
4          wähle ein zufälliges S' aus N_k(S)
5          wenn f(S') < f(S) dann
6              S ← S'
7              k ← 1
8          sonst
9              k ← k + 1
10     bis k = k_max
11 bis Abbruchkriterium erüllt ist
12 zurück S
```

RVNS ist vor allem für jene Probleme interessant, in denen Nachbarschaften zu groß sind, um ausführlich durchsucht zu werden, und eine lokale Suche auch zu kostspielig ist. Eine Liste von Problemen, für die sich das RVNS bewährt hat, ist in „Variable Neighbourhood Search: methods and applications“ [2] zu finden.

### 4.2.3 Basic Variable Neighborhood Search (BVNS)

Das BVNS erweitert das Prinzip des RVNS um eine lokale Suche. Die zufällig gewählte Lösung in Schritt 4 kann beliebig schlecht sein, aber trotzdem ganz nahe an der Optimallösung liegen. Bevor nun in die nächste Nachbarschaft gewechselt wird und man Lösungen untersucht, die ganz andere Eigenschaften haben, kann es von Vorteil sein, die nähere Umgebung der gewählten Lösung genauer zu untersuchen.

BVNS ist flexibel in Bezug auf die verwendete lokale Suche. Jede lokale Suche kann verwendet werden. Für jede Problemart kann so das beste bekannte Verfahren eingesetzt werden.

Der Vorteil aller VNS-Varianten ist, dass durch den Wechsel von Nachbarschaften das Verhalten der Suche beeinflusst wird. Die kleinen Nachbarschaften betrachten ähnliche Lösungen und helfen, lokale Optima zu finden. Die größeren Nachbarschaften helfen den lokalen Optima wieder zu entkommen und mehr vom Suchraum abzudecken. Das BVNS ist schneller als das VND, da das BVNS nicht immer die komplette Nachbarschaft absucht, und nicht ganz so zufällig wie das RVNS, da die lokale Suche zumindest eine kleine Nachbarschaft um eine Lösung absucht. BVNS kombiniert die Stärken von VND und RVNS.

**Algorithmus 4.4:** Basic Variable Neighborhood Search (BVNS)

**Eingabe** : eine Startlösung $S$, ein Set von Nachbarschaften $N_k$, wobei $k = 1, \ldots, k_{max}$, und ein Abbruchkriterium

**Ausgabe** : eine verbesserte Ausgangslösung, wenn Verbesserungen gefunden werden konnten, sonst die unveränderte Ausgangslösung

```
1  wiederhole
2      k ← 1
3      wiederhole
4          wähle ein zufälliges S' aus N_k(S)
5          suche eine lokale Verbesserung S'' von S'
6          wenn f(S'') < f(S) dann
7              S ← S''
8              k ← 1
9          sonst
10             k ← k + 1
11     bis k = k_max
12 bis Abbruchkriterium erfüllt ist
13 zurück S
```

Wie das BVNS Nachbarschaften wechselt, soll anhand der Abbildung 4.3 genauer erklärt werden. Es wird dazu nach dem Algorithmus 4.4 vorgegangen. Hierbei wird angenommen, dass es sich um ein Minimierungsproblem handelt und es das Ziel ist, die Lösung zu finden, für welche die Funktion $f$ den kleinsten Wert ergibt. Das BVNS startet von der Lösung $x$. Die erste Nachbarschaft $N_1(x)$ findet schon eine bessere Lösung $x'$. Es folgen mehrere Durchläufe der

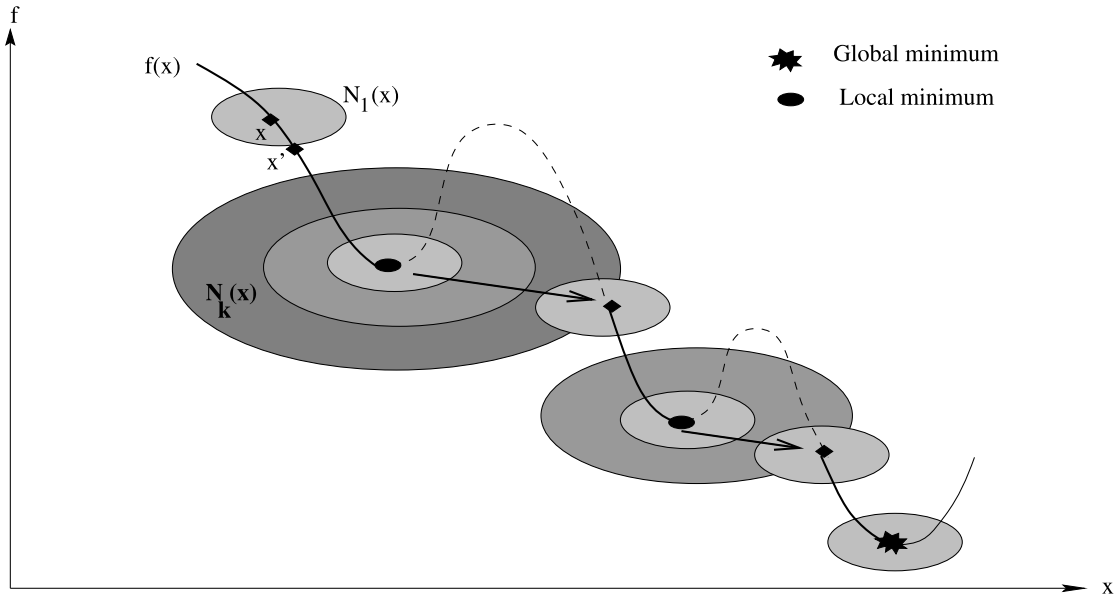



**Abbildung 4.3:** Visualisierung der Wechsel der Nachbarschaften beim BVNS (Abbildung aus [2])

zweiten Schleife (Zeile 3 bis 11), bevor ein lokales Optimum gefunden wird. Dies wird durch die gezogene Linie gekennzeichnet. In der kleinen Nachbarschaft des lokalen Optimums kann keine Verbesserung gefunden werden. Der Verlauf des $f$-Wertes wird durch die strichlierte Linie dargestellt. Erst in der $k$-ten Nachbarschaft wird eine bessere Lösung gefunden. Das erste lokale Optimum konnte damit überwunden werden. Mehrere kleine Verbesserungen in den kleinen Nachbarschaften führen zu einem neuen lokalen Optimum. Es handelt sich wieder um ein lokales Optimum, welchem nur durch Betrachten größerer Nachbarschaften entkommen werden kann. Die nächste Verbesserung führt schon zum globalen Optimum. Das BVNS kann jedoch nicht wissen, dass es am Ziel angekommen ist, und wird daher noch nach Verbesserungen suchen, bis das Abbruchkriterium erfüllt ist. Dieses Beispiel vernachlässigt, dass reale Probleme einen größeren und komplexeren Suchraum haben und oft sehr viele Versuche notwendig sind, um bessere Lösungen zu finden. Insbesondere einen Weg aus lokalen Optima hin zum globalen Optimum zu finden, ist sehr schwierig. Das Skewed VNS kann hier weiterhelfen.

### 4.2.4 Skewed Variable Neighborhood Search (SVNS)

Beim *Skewed Variable Neighborhood Search (SVNS)* handelt es sich um ein BVNS mit einem veränderten Akzeptanzkriterium. Es werden nicht nur verbessernde Lösungen akzeptiert, sondern auch schlechtere Lösungen, wenn sich die aktuelle Lösung und die schlechtere Lösung stark unterscheiden. Die Unterschiede zwischen Lösungen, auch deren Distanz $\rho(S', S'')$ genannt, muss für jedes Problem definiert werden. Die Distanz kann näherungsweise definiert werden über die Größe der Nachbarschaft, die notwendig ist, um von der aktuellen Lösung die

neue Lösung zu erzeugen. Doch auch diese Näherung ist nicht immer schnell und einfach zu bestimmen.

Die Distanz wird multipliziert mit einem Faktor $\alpha$. Je größer dieser Faktor, desto schlechtere Werte können Lösungen haben, und sie können doch noch akzeptiert werden. Ist der Faktor zu groß, wird jede Lösung übernommen, egal, wie schlecht sie ist. Welcher Wert sich für $\alpha$ gut eignet, hängt von Problem und Problemgröße ab und ist nur durch Experimente bestimmbar. Eine Pseudo-Code Darstellung des SVNS zeigt Algorithmus 4.5.

**Algorithmus 4.5:** Skewed Variable Neighborhood Search (SVNS)

**Eingabe** : eine Startlösung $S$, ein Set von Nachbarschaften $N_k$, wobei $k = 1, \ldots, k_{max}$, und ein Abbruchkriterium

**Ausgabe** : eine verbesserte Ausgangslösung, wenn Verbesserungen gefunden werden konnten, sonst die unveränderte Ausgangslösung

```
1  S* ← S
2  wiederhole
3      k ← 1
4      wiederhole
5          wähle ein zufälliges S' aus N_k(S)
6          suche eine lokale Verbesserung S'' von S'
7          wenn f(S'') < f(S*) dann
8              S* ← S ← S''
9              k ← 1
10         sonst wenn f(S'') − αρ(S'', S) < f(S) dann
11             S ← S'
12             k ← 1
13         sonst
14             k ← k + 1
15     bis k = k_max
16 bis Abbruchkriterium erfüllt ist
17 zurück S*
```

Die Idee hinter dieser Variante ist, dass auch andere Bereiche des Suchraums untersucht werden sollen, wenn die Umgebung eines lokalen Optimums ausreichend untersucht ist. Dazu müssen Lösungen generiert werden, die nicht viel mit der aktuellen Lösung gemein haben. Beim BVNS werden nur bessere Lösungen akzeptiert. Viele Bereiche des Suchraums werden dabei möglicherweise nie betrachtet, da nicht gleich eine bessere Lösung in diesen Bereich gefunden wird. So kann das Optimum unter Umständen nie gefunden werden. Ziel ist es aber, nicht jede (schlechte) Lösung zu untersuchen. Vermieden soll auch werden, viele Lösungen mehrmals zu untersuchen. Über die Distanzfunktion und den Multiplikationsfaktor wird gesteuert, dass nur entfernte Lösungen akzeptiert werden, also solche, die nicht gerade in der letzten lokalen Suche

und den letzten Nachbarschaften betrachtet wurden. Unterscheidet sich die neue Lösung von der aktuellen, so bekommt die neue Lösung einen Bonus bei der Evaluierung. Die neue Lösung kann etwas schlechter sein als die aktuelle und trotzdem noch übernommen werden. Wie groß die Verschlechterung sein darf, hängt von der Distanz und dem Faktor $\alpha$ ab.

Der Nachteil dieses Verfahrens ist, dass die Distanzfunktion oft schwierig zu definieren und zu berechnen ist. Andere Akzeptanzkriterien schneiden dadurch in der Praxis manchmal besser ab [3, 12].

### 4.2.5 Variable Neighborhood Search mit Simulated Annealing als Akzeptanzkriterium (VNS-SA)

In diesem Kapitel wird eine Variante des VNS mit SA als Akzeptanzkriterium beschrieben. Das eingesetzte VNS baut auf dem BVNS auf und verwendet die Grundidee des SVNS. Um die Schwierigkeiten des SVNS zu umgehen, wurde SA als Akzeptanzkriterium eingesetzt. Diese Variante ist angelehnt an die Arbeit von V. Hemmelmayr et al. [3] (siehe auch Kapitel 3.5). Es wurden jedoch nicht alle Nachbarschaften übernommen, eine andere Art der lokalen Suche verwendet und eine nicht-monotone Abkühlungsstrategie für die SA-Heuristik verwendet. Es folgt eine genauere Beschreibung des Verfahrens unter Berücksichtigung der Unterschiede zu [3]. Algorithmus 4.6 zeigt die Pseudo-Code-Auflistung des implementierten VNS.

#### 4.2.5.1 Eingesetzte Nachbarschaften

Es wurde ein Set von 15 Nachbarschaften ($k_{max} = 15$) gewählt. Bei den Nachbarschaften $N_1$ bis $N_{10}$ werden die Besuchsmuster der Kunden verändert. Die Nachbarschaften $N_{11}$ bis $N_{15}$ verändern die Touren der Fahrzeuge. Segmente einer Tour eines Fahrzeuges werden zu einem anderen Fahrzeug transferiert. Die Größe der Nachbarschaften wird über Parameter gesteuert. Tabelle 4.1 gibt eine Übersicht über die gewählten Parameter.

Beim PVRP ergibt sich für jeden Kunden durch die Besuchsfrequenz eine Anzahl von Besuchsmustern. In dem hier zu lösenden PVRP werden alle zulässigen Besuchsmuster bereitgestellt. Beim Konstruieren der ersten Lösung wurden die Besuchsmuster zufällig gewählt (siehe Kapitel 4.1). Am wichtigsten ist es daher, eine gute Aufteilung der Kunden auf den Planungszeitraum zu erreichen. Der erste Operator, welcher die Nachbarschaften $N_1$ bis $N_{10}$ definiert, weist den Kunden ein neues Besuchsmuster zu. Über einen Parameter kann angegeben werden, für wie viele Kunden das Besuchsmuster verändert werden soll. Die Kunden werden per Zufall ausgewählt. Jeder der ausgewählten Kunden wird aus allen Touren entfernt. Danach wird ein neues Besuchsmuster ausgewählt und der Kunde entsprechend dem ausgewählten Besuchsmuster wieder in Touren eingefügt. Das Einfügen eines Kunden in eine Tour erfolgt durch *Best Insertion (BI)*. Bei BI werden alle Fahrzeuge und jede mögliche Position in den Touren der Fahrzeuge ausprobiert, und der Kunde wird dort eingefügt, wo die geringsten Zusatzkosten entstehen. Dieses Vorgehen ist zeitintensiv, liefert aber bessere Lösungen als andere Einfügeheuristiken.

Der Move-Operator versucht, bessere Zuweisungen von Kunden zu Fahrzeugen zu finden. Dazu wird ein Segment mit Kunden aus einer Tour eines Fahrzeugs entfernt und einem anderen Fahrzeug hinzugefügt, wie in Abbildung 4.4 zu sehen ist. Die maximale Länge des Segments wird durch einen Parameter des Operators festgelegt. Die tatsächliche Länge wird per Zufall

**Algorithmus 4.6:** Variable Neighborhood Search mit Simulated Annealing als Akzeptanzkriterium (VNS-SA)

**Eingabe** : eine Startlösung $S$, ein Set von Nachbarschaften $N_k$, wobei $k = 1, \ldots, k_{max}$, eine Anfangstemperatur $t_{init}$ und ein Abkühlungsfaktor $\alpha$ für das Simulated Annealing und ein Abbruchkriterium

**Ausgabe** : eine verbesserte Ausgangslösung, wenn Verbesserungen gefunden werden konnten, sonst die unveränderte Ausgangslösung

```
1  S* ← S
2  t ← t_init
3  i ← 0
4  wiederhole
5      k ← 1
6      wiederhole
7          wähle ein zufälliges S' aus N_k(S)
8          suche eine lokale Verbesserung S'' von S'
9          repariere S''
10         wenn f(S') < f(S*) dann
11             S* ← S ← S'
12             k ← 1
13             i ← 0
14         sonst wenn f(S') < f(S) dann
15             S ← S'
16             k ← 1
17         sonst wenn e^{-|f(S')-f(S)|/t} > Zufallszahl() dann
18             S ← S'
19             k ← 1
20         sonst
21             k ← k + 1
22         t ← t · α
23     bis k = k_max
24     i ← i + 1
25 bis i = i_max
26 zurück S*
```

| $k$ | Operator | Max. Kunden |
|---|---|---|
| 1 | Wechsle Besuchsmuster | 1 |
| 2 | Wechsle Besuchsmuster | 2 |
| 3 | Wechsle Besuchsmuster | 3 |
| 4 | Wechsle Besuchsmuster | 4 |
| 5 | Wechsle Besuchsmuster | 5 |
| 6 | Wechsle Besuchsmuster | 6 |
| 7 | Wechsle Besuchsmuster | 7 |
| 8 | Wechsle Besuchsmuster | 8 |
| 9 | Wechsle Besuchsmuster | 9 |
| 10 | Wechsle Besuchsmuster | 10 |

| $k$ | Operator | Min. Segmentlänge | Max. Segmentlänge |
|---|---|---|---|
| 11 | Move | 1 | min(1, n) |
| 12 | Move | 1 | min(2, n) |
| 13 | Move | 1 | min(3, n) |
| 14 | Move | 1 | min(4, n) |
| 15 | Move | 1 | min(5, n) |

**Tabelle 4.1:** Set der Nachbarschaften

bestimmt, genauso wie der Startpunkt des Segments und die beteiligten Fahrzeuge. Auch die Position, an der das Segment eingefügt wird, ist zufällig. Es wird keine Einfügeheuristik, wie z.B. Best-Insertion, verwendet. In Experimenten hat sich gezeigt, dass die lokale Suche in fast allen Fällen bessere Lösungen finden konnte, wenn die Kunden des eingefügten Segments separat in die bestehende Tour eingegliedert werden. Da die Kunden getrennt werden, kann eine spezielle Einfügeheuristik eingespart werden.

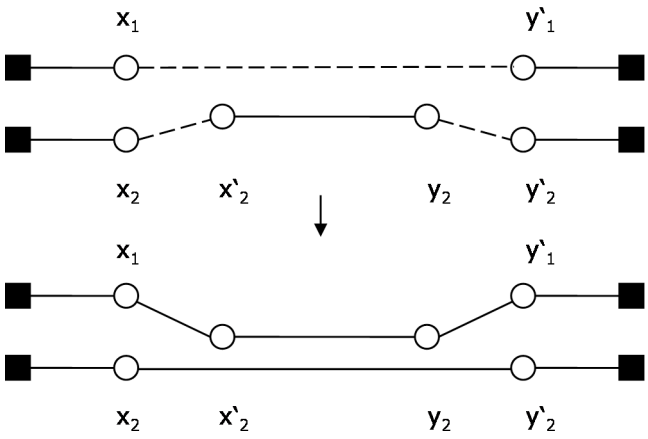



**Abbildung 4.4:** MOVE-Operator (Abbildung aus [3])

In [3] wurde auch noch ein Cross-Operator verwendet. Dieser verschlechterte das Laufzeitverhalten des Algorithmus, aber es wurden keine besseren Lösungen für das hier zu lösende PVRP der Müllentsorgung gefunden. Der Operator wurde daher wieder aus dem Set der verwendeten Nachbarschaften entfernt.

#### 4.2.5.2 Verwendete Local Search (LS)

Die Nachbarschaften suchen nach besseren Lösungen durch das Verändern von Besuchsmustern und das Verschieben von Kunden von einem Fahrzeug zu einem anderen. Durch das Entfernen und Einfügen von Kunden aus den Touren werden die Touren in den meisten Fällen ineffizienter. In den Touren der Fahrzeuge liegt daher großes Einsparungspotential. Die lokale Suche soll durch Permutieren der Besuchsreihenfolge die Tourlänge für die Fahrzeuge verkürzen.

Eine der am häufigsten eingesetzten Verbesserungsheuristiken für das VRP ist das 2-Opt. Bei diesem Verfahren handelt es sich um eine Spezialisierung des $k$-Opt [15]. Dieses ist in der Praxis erprobt und gut verstanden. $k$-Opt ist deterministisch und kommt bei den meisten Instanzen in kurzer Zeit zu guten Ergebnissen. Nur in einigen Fällen, welche in der Praxis extrem selten vorkommen, kann eine exponentielle Anzahl von Schritten notwendig sein, bevor der Algorithmus terminiert. Wird ein großes $k$ gewählt, so sind die gefunden Lösungen besser. Dieser Vorteil wird mit einer längeren Laufzeit erkauft. Es ist daher wichtig, den richtigen Trade-Off zwischen Lösungsqualität und Rechenzeit zu finden. Aus der Arbeit von Hemmelmayr et al. zum PVRP [3] geht hervor, dass es wichtiger ist, die richtigen Besuchsmuster und gute Zuordnungen zu Fahrzeugen als in jeder Lösung die optimale Tour für ein Fahrzeug zu finden. Daher kommt als lokale Suche das 2-Opt zum Einsatz und nicht wie bei [3] eine Version des 3-Opt. So können Verbesserungen schneller gefunden werden, und es bleibt mehr Zeit, um weitere Nachbarschaften zu untersuchen.

Das 2-Opt betrachtet jede Tagestour der Fahrzeuge einzeln. Zwei Kanten werden aus einer Tour entfernt und zwei neue Kanten eingefügt (siehe Abbildung 4.5). Dabei können drei Fälle eintreten [1, 4]:

1. Es werden dieselben Kanten eingefügt, die entfernt wurden. Da sich die Tour dabei nicht ändert, ist dieser Fall zu vermeiden.

2. Die zwei neuen Kanten führen zu zwei getrennten Touren. Da nur zusammenhängende Touren gefahren werden können, führt dieser Fall zu ungültigen Lösungen.

3. Durch die zwei neuen Kanten entsteht eine neue Tour, in welcher die Kunden eines Teilstücks in umgekehrter Reihenfolge besucht werden (siehe Abbildung 4.5).

Nur der dritte Fall kann zu neuen zulässigen Lösungen führen, die möglicherweise auch kürzer sind. Es werden lediglich jene Austauschoperationen übernommen, welche zu einer gültigen Lösung führen und eine Verbesserung bringen. Erst wenn durch das Auswechseln zweier Kanten keine Verbesserungen mehr gefunden werden, terminiert das 2-Opt-Verfahren. Eine solche Lösung wird als 2-optimal bezeichnet [4].

Das 2-Opt versucht, die benötigte Fahrzeit zu reduzieren, dabei wird aber die Kapazitätsbeschränkung des Fahrzeugs nicht beachtet. Das bedeutet, dass die Lösungen, die das 2-Opt liefert,

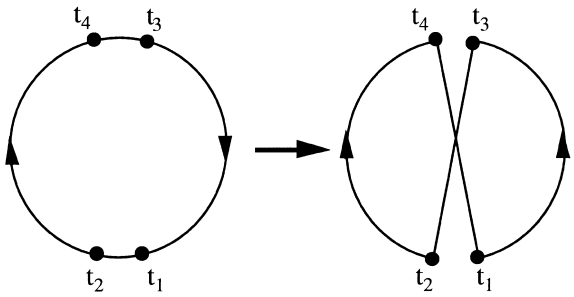



**Abbildung 4.5:** 2-Opt-Austausch (Abbildung aus [4])

nicht immer alle Nebenbedingungen erfüllen. Daher werden nach jedem 2-Opt die Lösungen repariert. Dabei wird eine IF eingefügt, sobald ein Fahrzeug voll ist. Je nachdem, ob das Fahrzeug in leerem Zustand ins Depot zurückfahren muss oder nicht, werden auch hier noch IF angefahren. Welche IF besucht wird hängt vom gewählten IF-Modus ab. Es wird jene IF gewählt, die am nächsten zu den beiden Kunden liegt, zwischen denen sie eingefügt wird, und welche alle weiteren Bedingungen (z.B.: prozentmäßiger Anteil) noch erfüllt. Sind auch Zeitfenster zu berücksichtigen, so werden all jene Kunden, für die das Zeitfenster überschritten wird, an den Anfang der Tour verschoben. Dies garantiert jedoch nicht, dass alle Zeitfenster eingehalten werden.

Das 2-Opt Verfahren ist ein wichtiger Bestandteil des VNS und findet oft bessere Lösungen, als durch die BI-Heuristik und den Move-Operator entstehen.

#### 4.2.5.3 Akzeptanzkriterium - Simulated Annealing (SA)

Die Idee des veränderten Akzeptanzkriteriums entstammt dem SVNS. Beim PVRP lässt sich die Distanz zwischen zwei Lösungen nur aufwändig bestimmen. Weiteres haben Versuche gezeigt, dass andere Akzeptanzkriterien besser abschneiden [3]. Simulated Annealing hat um $2.71\%$ bessere Ergebnisse erzielt als SVNS und schneidet auch besser ab als Treshhold Accepting [12].

Bei SA handelt es sich um eine der ersten Metaheuristiken, welche Kirkpatrick et al. 1983 entwickelt haben [17]. Sie ist dem Verhalten von Kristallen und Metallen nachempfunden. Diese ändern im erhitzten Zustand leicht ihre Form. Beim Abkühlen verlieren sie diese Eigenschaft und es ist schwerer, ihre Form zu ändern. Das SA startet mit einer bestimmten Temperatur und ändert diese im Verlauf der Suche. Ist die Temperatur hoch, steigt die Wahrscheinlichkeit, dass auch schlechtere Lösungen übernommen werden. Verbessernde Lösungen werden immer akzeptiert. Sinkt die Temperatur, sinkt auch die Wahrscheinlichkeit, schlechtere Lösungen zu akzeptieren. In dieser Phase wird im Bereich der aktuellen Lösung intensiver gesucht. Das SA hat zwei Parameter, welche die Performanz der Suche maßgebend beeinflussen. Das sind die Starttemperatur $t_{init}$ und die Abkühlungsstrategie. Die am häufigsten genutzte Strategie ist das geometrische Abkühlen. Um die Temperatur für die nächste Iteration zu erhalten, wird die aktuelle Temperatur mit einem Faktor $\alpha \in (0, 1)$ multipliziert. Man spricht von nicht-monotonen

Abkühlungsstrategien, wenn sich Phasen der Abkühlung mit Phasen der Erhitzung abwechseln.

Für das PVRP wurde das BVNS mit dem SA erweitert, sodass bessere Lösungen immer akzeptiert werden und schlechtere nur, wenn das SA sie akzeptieren würde. Als Starttemperatur wurde $5.0$ gewählt, und $\alpha$ wurde gleich $0.95$ gesetzt. Wenn für 500 Iterationen das BVNS keine bessere Lösung findet, dann wird die Temperatur wieder auf den Startwert gesetzt. Dadurch wird es nach ausreichender Intensivierung der Suche auf einen bestimmten Bereich wieder möglich, andere Bereiche des Suchraums zu betrachten. Man kann daher sagen, dass es sich um eine nicht-monotone Abkühlungsstrategie handelt.

KAPITEL 5

# Umsetzung

Es gilt, das Problem von Abfallentsorgungsunternehmen zu lösen. Diese Unternehmen suchen nach guten Touren für ihre Müllsammelfahrzeuge. Dazu wurde ein Verfahren entwickelt, welches solche Probleme löst. Wie die Schnittstelle zum Optimierungsmodul aussieht wird nun am Anfang dieses Kapitels erklärt. Anschließend wird genauer auf Implementierungsdetails eingegangen.

Das Lösen eines Problems kann mehrere Stunden dauern. Die Laufzeit hängt dabei direkt von der Problemgröße, d.h. von der Anzahl der Kunden und Fahrzeuge und den gewählten Parametern ab. Zum Beispiel verursachen Zeitfenster einen Mehraufwand beim Berechnen der Kosten. Das Lösen eines Problems mit Zeitfenstern wird daher mehr Zeit benötigen.

Schnelle Prozessoren und viel Arbeitsspeicher können die Laufzeit verkürzen. Um eine hohe Auslastung der Hardware zu erreichen, wird die Optimierung nur auf einem Rechner, dem Server, durchgeführt. Client-Anwendungen können sich mit dem Server verbinden und dem Server Probleme zum Lösen übermitteln. Für jedes Problem erstellt der Server einen Solver. Jeder Solver läuft in einem eigenen Prozess und ist für das Lösen genau eines Problems zuständig. Über den Server können Client-Anwendungen Befehle an einen Solver schicken.

Ist ein Solver einmal erstellt, muss ein Client den Solver starten. Erst dann beginnt der Solver mit der Arbeit. Der Solver sucht nach besseren Lösungen und stoppt erst, wenn für 1000 Iterationen keine Verbesserung mehr gefunden wurde. Der aktuelle Status des Solvers kann immer erfragt werden. Eine Client-Anwendung kann eine Lösung verlangen, auch während der Solver noch nach Verbesserungen sucht. Der Solver liefert dann die beste bisher gefundene Lösung. Hat die retournierte Lösung die gewünschte Güte oder steht nicht mehr Zeit zur Verfügung, kann eine Client-Applikation den Solver jederzeit stoppen.

Die Optimierung beginnt mit der Erstellung einer Startlösung. Diese ermittelt der Savings-Algorithmus. Mit der Startlösung wird dann das VNS gestartet. Die Nachbarschaften des VNS ändern die Besuchsmuster und Routen der Fahrzeuge. Die lokale Suche optimiert für alle Fahrzeuge alle Tagesrouten. Findet das VNS für 1000 Iterationen keine Verbesserung, so wird die Arbeit eingestellt.

## 5.1 Datenformat

Daten werden in Form von *Extensible Markup Language-Dokumenten (XML)* zwischen Client und Server ausgetauscht. Eine möglichst reibungslose Kommunikation soll durch das Definieren einer Struktur für die Daten erreicht werden. Die Struktur wird in diesem Kapitel beschrieben.

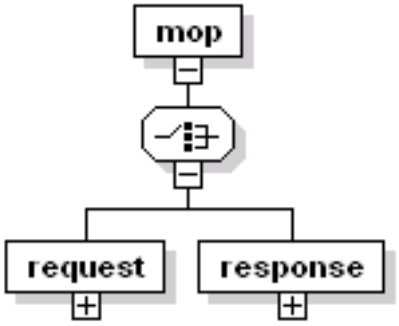



**Abbildung 5.1:** Request-Response-Modell der Müllentsorgungsoptimierung (MOP)

Eine Anfrage kann an den Server oder an einen Solver gerichtet sein. Ein Server akzeptiert zwei Arten von Anfragen:

- *shutdown*: Weist den Server an, herunterzufahren. Der Server bestätigt diesen Befehl. Dann stoppt der Server alle laufenden Optimierungen und verwirft alle Resultate.
- *status*: Sagt dem Server, er soll melden, welche Solver er gerade verwaltet. Als Antwort liefert der Server eine Liste mit allen Solvern und deren Status.

Ein Solver ist über eine ID identifizierbar. So kann dieser, auch nach der Erstellung eines Solvers, folgende Befehle empfangen:

- *create*: Hat der Solver die ID *0*, so erstellt dieser Befehl einen neuen Solver. In allen anderen Fällen wird dieser Befehl ignoriert. Die ID des erstellten Solvers wird als Antwort auf diese Anfrage zurückgesandt.
- *start*: Der Solver beginnt, nach einer Lösung zu suchen.
- *stop*: Der Solver bricht seine Arbeit ab. Die berechnete Lösung bleibt bis zur Löschung des Solvers erhalten.
- *status*: Der Solver liefert seinen aktuellen Status zurück.
- *result*: Vom Solver wird das aktuelle Ergebnis zurückgeschickt.
- *dispose*: Veranlasst die Löschung des Solvers und der berechneten Lösung.

Für alle Befehle gilt, dass sie zu jeder Zeit empfangen und verarbeitet werden. Die Befehle *status* und *result* liefern Ergebnisse, auch während der Solver optimiert.

Direkt nach seiner Erstellung ist der Status des Solvers *idle*. Empfängt er den *start*-Befehl, so wechselt er seinen Status auf *working*. Bekommt er die Aufforderung, seine Arbeit zu stoppen, so ist er wieder *idle* und kann mit dem *start*-Befehl veranlasst werden, seine Arbeit von neuem zu beginnen. Konnte er seine Arbeit abschließen, so meldet er *finished*. Die Statusmeldung *disposed* ist die Antwort auf den *dispose*-Befehl. Tritt ein Fehler auf, so meldet der Solver *error* und eine Beschreibung des Fehlers.

Eine Antwort ist, genau wie eine Anfrage, aufgeteilt in einen Bereich für den Server und in einen für den Solver.

Alle Anfragen werden durch eine Antwort mit denselben Elementen bestätigt. Im Fall von *status* und *result* beinhalten die Antworten noch die angeforderten Informationen.

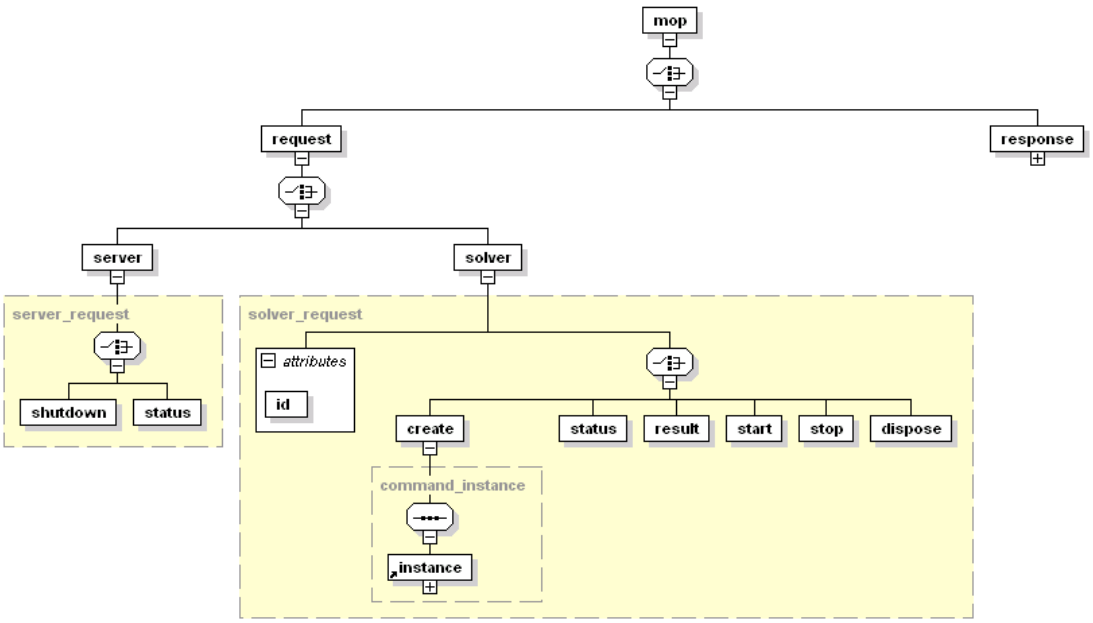



**Abbildung 5.2:** Aufbau einer Anfrage an den Server

In einer *create*-Anfrage enthält ein *instance*-Element die Daten des zu lösenden Problems. Als Antwort auf eine *result*-Anfrage beinhaltet ein *instance*-Element auch noch die Lösung des Problems. D.h. ein *instance*-Element enthält auf jeden Fall die problemspezifischen Input-Daten und kann, wenn es sich um ein Ergebnis eines Solvers handelt, auch die Lösung des Problems liefern. Das *instance*-Element ist Container für folgende Daten:

- *distances*: Dieses Element speichert für jeden Ort die Distanz zu allen anderen Orten. Die vergebene *location id* ist eindeutig und entspricht der ID von Kunden, Depots und IFs.

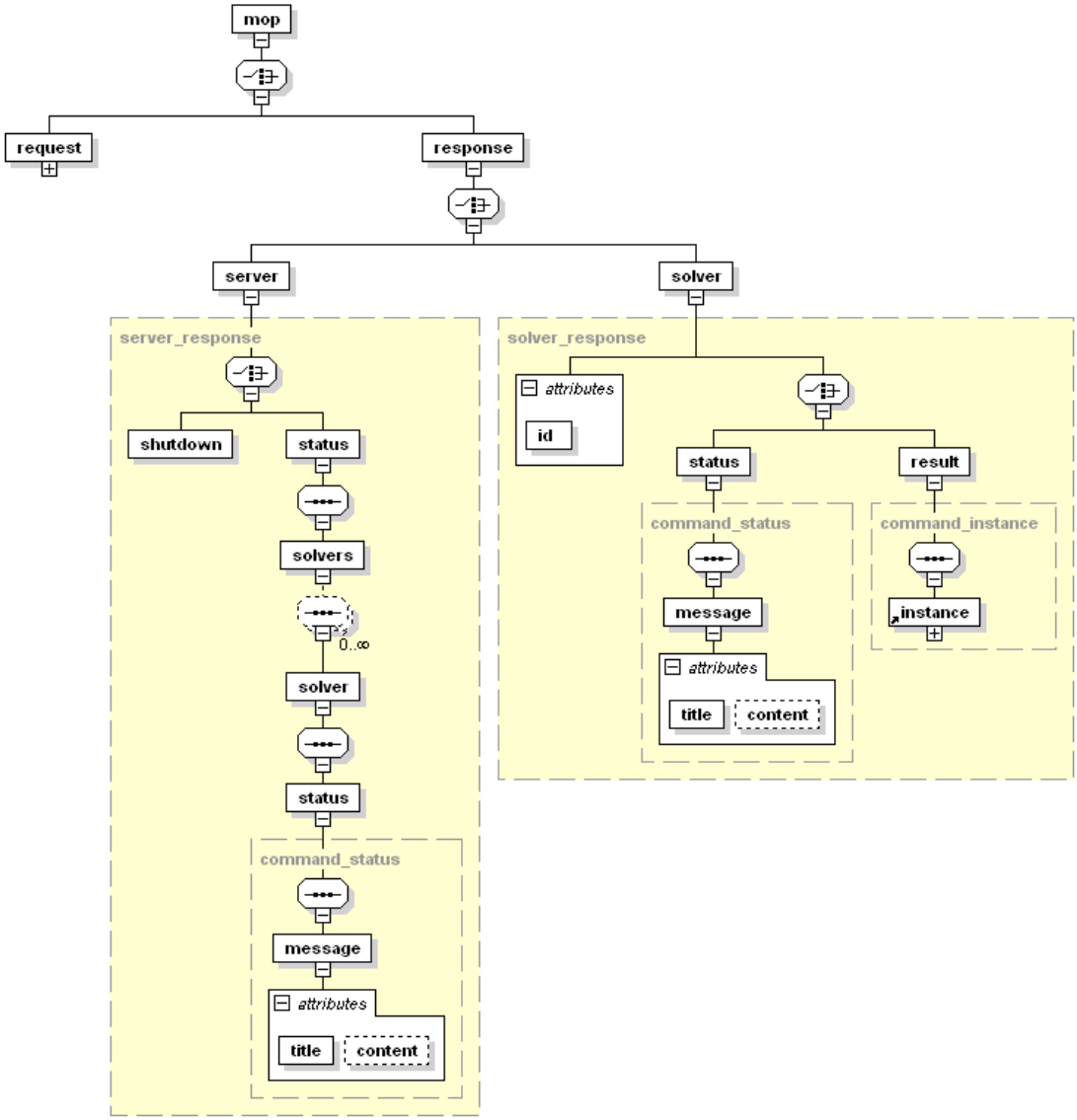



**Abbildung 5.3:** Aufbau einer Antwort des Servers

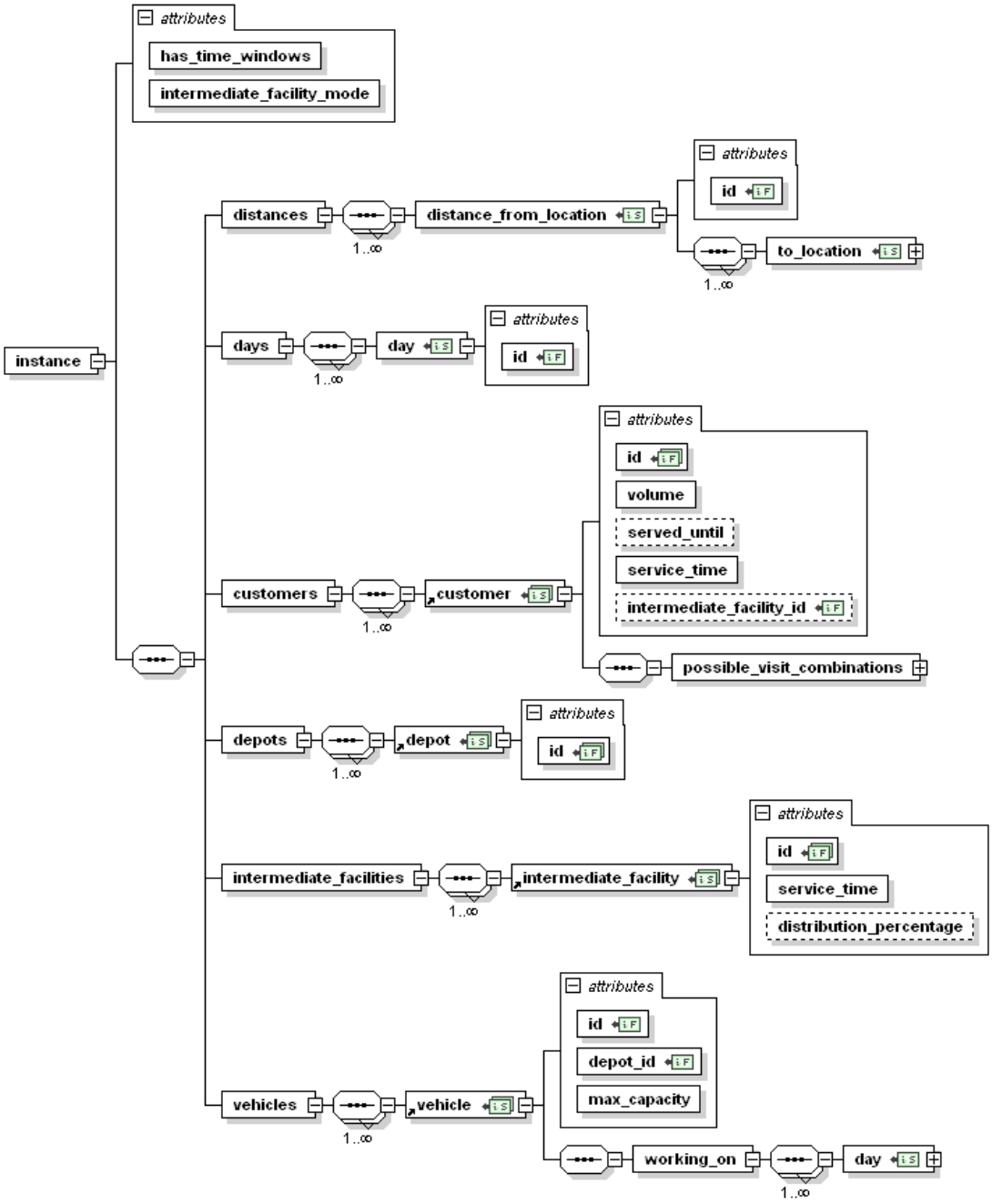



**Abbildung 5.4:** Struktur eines *instance*-Elements

- *days*: Die Optimierung soll für einen gewissen Planungszeitraum stattfinden. Dieses Element enthält jene Tage, die in diesen Planungszeitraum fallen. Die Tage sind, beginnend bei null, aufsteigend durchnummeriert.
- *customers*: Die zu besuchenden Kunden und deren Attribute befinden sich in dieser Auflistung.
- *depots*: Enthält alle möglichen Depots, von denen Fahrzeuge starten.
- *intermediate facilities*: Hier sind alle IFs vermerkt, zu denen Müll gebracht werden kann.
- *vehicles*: Die verfügbaren Fahrzeuge sind in diesem Element gespeichert.
- *has time windows*: Dieser Wert kann *true* oder *false* sein und gibt an, ob Zeitfenster bei der Optimierung berücksichtigt werden müssen.
- *intermediate facility mode*: Zu welcher Intermediate Facility ein Fahrzeug fährt, hängt vom Wert dieses Attributs ab. Es stehen *fixed*, *free*, *mixed* und *percentage* zur Verfügung.

Für *customer*, *depot* und *intermediate facility* gilt, dass deren ID auf eine ID in der Distanzmatrix referenziert. Dadurch sind genannte Elemente nicht nur unter ihresgleichen eindeutig identifizierbar, sondern auch untereinander. Die IDs von *0* bis *c* beziehen sich auf Kunden, *c+1* bis *d* auf Depots und *d+1* bis *i* auf IFs.

Ein *customer*-Element enthält als Attribute seine ID, sein Müllvolumen, seine Service-Zeit und als Sequenz noch die möglichen Besuchsmuster. Ein Besuchsmuster besteht aus einer Auflistung von Tagen. Umfasst das zu lösende Problem Zeitfenster, so ist auch noch für das Attribut *served until* ein Wert notwendig. Bei der fixen Intermediate Facility Zuordnung ist in dem Feld *intermediate facility id* die ID der Intermediate Facility vermerkt, zu der der Müll dieses Kunden gebracht werden muss.

IFs werden durch deren ID und die Service-Zeit beschrieben. Ist bei dem Problem der Zuordnungsmodus für IFs mit *percentage* angegeben, so ist der Prozentsatz im Attribut *distribution percentage* gespeichert.

Ein Depot besteht nur aus seiner ID.

Ein *vehicle*-Element speichert neben einer Fahrzeug-ID auch noch die ID seines Depots (*depot id*) und die maximale Ladekapazität (*max capacity*). Da ein Fahrzeug nicht immer verfügbar sein muss, ist für jeden Tag anzugeben, wie viele Stunden es eingesetzt werden kann (*max duration*) und ob es beladen ins Depot zurückfahren darf oder vorher die Ladung löschen muss (*return empty* ist *true* bzw. *false*). Kommen Zeitfenster zum Einsatz, so ist auch eine Startzeit (*start time*) mitzuliefern. Ein *vehicle*-Element entspricht genau einem realen Fahrzeug. Ein Fahrzeug kann jedoch mehreren *vehicle*-Elementen zugeordnet sein. Das kann notwendig sein, wenn zum Beispiel an einem Tag eine Unterbrechung der Tour notwendig ist, danach jedoch wieder weitergefahren werden soll. Dies kann dann über zwei *vehicle*-Elemente bewerkstelligt werden.

Eine Lösung eines Problems liefert für jeden Einsatztag (*day* in *working on*) eines *vehicle*-Elements die Gesamtkosten (*cost*), die Fahrzeit (*duration*) und die Tour des Fahrzeugs (*tour*). Eine Tour besteht aus Stopps (*stop*). Die Reihenfolge, in der die Orte (*location id*) besucht werden, ergibt sich durch das *position*-Attribut. Der erste Stopp hat die Position *0*, der zweite Stopp die Position *1* usw. .

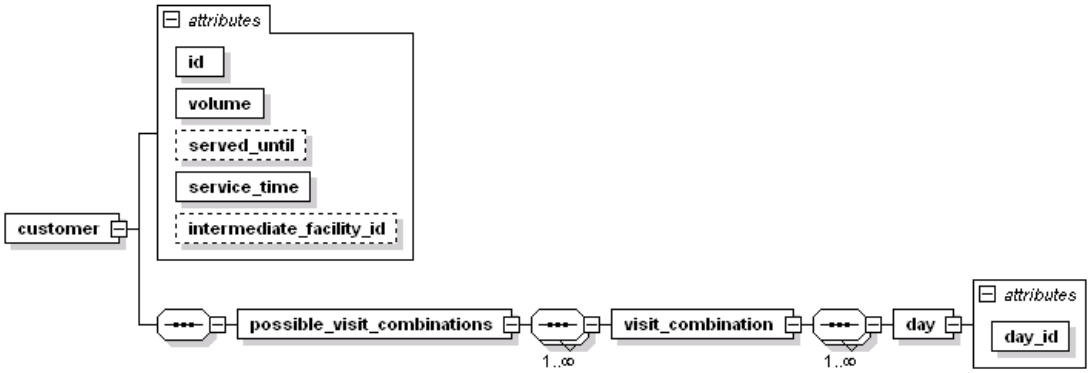



**Abbildung 5.5:** Struktur eines *customer*-Elements

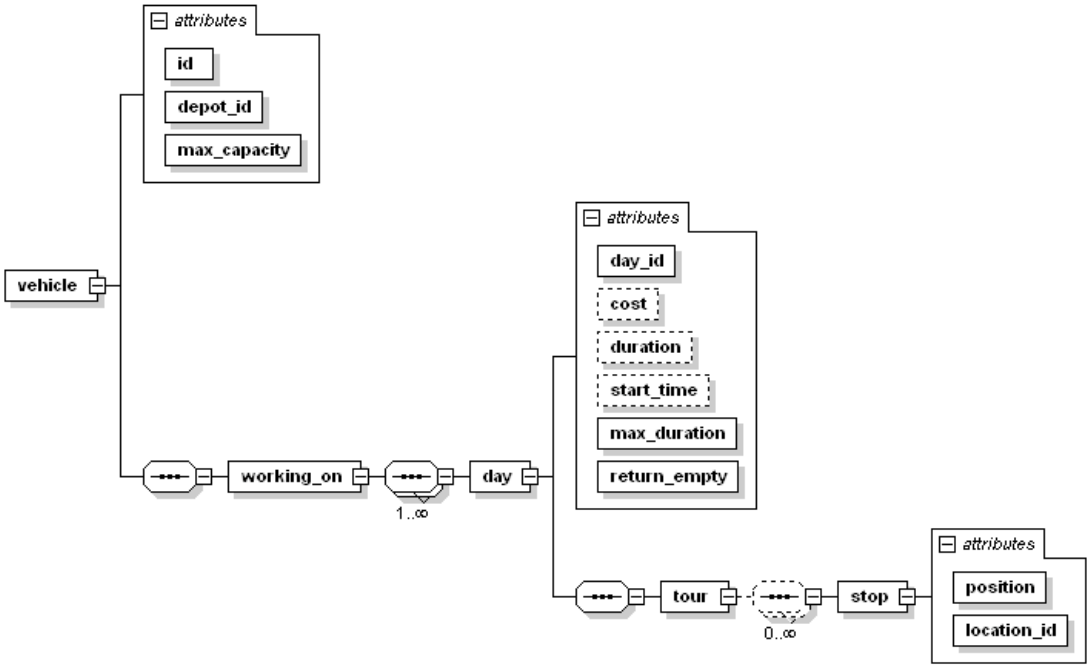



**Abbildung 5.6:** Struktur eines *vehicle*-Elements

## 5.2 Datenstrukturen

Nachdem die Daten empfangen wurden, müssen sie aufbereitet werden, damit die Lösungsalgorithmen schnell die richtigen Informationen abfragen können. Die Performance des Optimierungsmoduls hängt zu einem großen Teil von den gewählten Strukturen für die Daten ab. Deshalb wurden besonders effiziente Strukturen für folgende Problem- bzw. Lösungsdaten gewählt:

- **Distanzmatrix**: Die Distanzmatrix enthält die Entfernungen der einzelnen Standorte zueinander. Die Fahrzeit einer Tour berechnet sich aus der Summe der Distanzen zweier aufeinanderfolgender Stopps. Repräsentiert wird die Distanzmatrix als ein zwei-dimensionales Array aus Integer-Zahlen.

- **Knoten**: Jeder Kunde, jedes Depot und jede Intermediate Facility werden als Knoten gespeichert. Knoten enthalten jene Informationen, die für alle drei Typen gleich sind: ID, Typ, Müllvolumen, Servicezeit. Die typenspezifischen Informationen werden seltener gebraucht und sind in den Klassen *Customer*, *Depot* und *Intermediate Facility* gespeichert. Durch die Abtrennung der selten gebrauchten Informationen kann auf die allgemeinen Informationen schneller zugegriffen werden.

- **Fahrzeug-Flotte**: Eine Fahrzeug-Flotte kann aus beliebig vielen heterogenen Fahrzeugen bestehen, d.h. es gibt eine Liste von Fahrzeugen. Jedes Fahrzeug sammelt die Touren, die es jeden Tag fährt. Ein Fahrzeug hat daher eine Liste von Touren. Eine Tour ist wiederum eine Liste von Kunden und IFs. Da das Depot für alle Touren gleich ist und immer am Anfang und am Ende vorkommen muss, wird es nicht in jeder Tour gespeichert, sondern nur einmal für das ganze Fahrzeug. Dasselbe gilt für die Kapazität.

## 5.3 Algorithmus

Der Algorithmus durchsucht sehr viele verschiedene Lösungen. Nur ein kleiner Prozentsatz der neuen Lösungen verbessert die beste bisher bekannte Lösung. Viele Lösungen werden erstellt, ihre Kosten ermittelt und dann wieder verworfen. Die Lebensdauer einer Lösung ist daher meistens sehr kurz.

Für die Laufzeit des Verfahrens ist es deshalb essentiell, dass das Verändern und das Evaluieren von Lösungen in kurzer Zeit möglich sind.

Das Kopieren einer Lösung benötigt viel Zeit und ist in Relation zur Lebensdauer unverhältnismäßig zeitintensiv, nur um zu überprüfen, ob Änderungen die Lösung verbessern oder verschlechtern. Daher wird die Lösung nicht kopiert, sondern eine Änderungsliste angelegt. In der Änderungsliste wird vermerkt, wie jede Änderung wieder rückgängig gemacht werden kann. Sollten sich Änderungen als nicht vorteilhaft erweisen, können sie wieder rückgängig gemacht werden. Dies benötigt weniger Zeit, als die Lösung zu kopieren.

Die Evaluierung einer Lösung benötigt nicht viel Zeit, wenn sie nur einmal ausgeführt wird. Da sie jedoch sehr oft aufgerufen wird, ist sie in Summe für einen großen Teil der Gesamtlaufzeit des Verfahrens verantwortlich.

Oft verändern sich nur Teile der Lösung. Eine Evaluierung muss in diesem Fall nicht alles neu berechnen, sondern nur die Bestandteile der Lösung, die sich verändert haben. Dazu müssen die Kosten der Bestandteile der Lösung gespeichert und es muss vermerkt werden, sobald diese nicht mehr aktuell sind. Dadurch ist etwas mehr Speicher notwendig, wobei die Laufzeitverbesserungen diesen Mehraufwand wieder wettmachen.

## 5.4 Implementierung

Bei der Implementierung wurde vor allem auf die Wiederverwendbarkeit Wert gelegt. In späteren Ausbaustufen soll das Optimierungsmodul für verschiedene Entsorger verwendet werden, und es ist davon auszugehen, dass sich deren Probleme nicht immer nur in der Problemgröße unterscheiden werden. Es sind daher die Programmiersprache und die Architektur mit bedacht gewählt worden, sodass auch bei neuen Problemen das Optimierungsmodul schnell und einfach angepasst werden kann.

Schon beim Schreiben der Spezifikation war absehbar, dass das Optimierungsmodul von einer Person geschrieben und später von anderen Personen gewartet und weiterentwickelt wird. Auch die Einbindung des Optimierungsmoduls wird wieder von anderen Personen vorgenommen. Von oberster Priorität waren daher eine gute Dokumentation und eine leichte Verständlichkeit des Programmcodes. Letztere hängt zu einem großen Teil auch von der gewählten Programmiersprache ab. Da die zu lösenden Probleme strategischer Natur und nicht zeitkritisch sind, ist die Performance nachrangig.

Als Programmiersprache für die Umsetzung standen C++ und Python zur Wahl. Erstere ist weit verbreitet und viele Programmierer glauben, C++ zu beherrschen. Viele Programme, bei denen die Laufzeit von Bedeutung ist, werden in C++ geschrieben.

Bei Python wurde schon beim Sprachdesign spezielles Augenmerk auf die Lesbarkeit des Quellcodes gelegt. Das Sprachdesign von Python macht Python-Programme in vielen Fällen aber auch langsamer als entsprechende C++-Programme. Aus Erfahrungen mit früheren Optimierungsprojekten ist bekannt, dass nicht alle Bestandteile eines Optimierungsmoduls zu gleichen Teilen an der Laufzeit beteiligt sind. Da es sich nur um einen kleinen Teil handelt, welcher bei der Optimierung die meiste Zeit benötigt, wurde die Entscheidung getroffen, eine Kombination von Python und Cython einzusetzen. Bei Cython handelt es sich um ein Framework, mit dem sich C-Erweiterungen für Python in einer Python ähnlichen Sprache schreiben lassen, welche aber in C-Code übersetzt werden und so eine ähnliche Leistung wie C und C++ erbringen. Nur jene Stellen, welche öfter ausgeführt werden oder lange Zeit benötigen, wurden mit Cython umgesetzt.

KAPITEL 6

# Tests und Resultate

Dieses Kapitel beschäftigt sich mit dem Testen der Implementierung. Im Folgenden wird beschrieben, mit welchen Daten getestet wurde, welche Hardware Verwendung fand und welche Resultate die Tests erbrachten.

## 6.1 Verwendete Testinstanzen

Getestet wurde die Implementierung mit Problemen unterschiedlicher Größe und Parameter. Die Testinstanzen bilden die Situation einer mittelgroßen Stadt in Österreich ab und wurden von einem Entsorgungsunternehmen zur Verfügung gestellt. Da bei diesen Daten nicht nur Müllsammelstellen, sondern auch Firmen berücksichtigt werden, wird im Folgenden allgemein von Kunden gesprochen.

In Abbildung 6.1 ist eine Karte der Stadt zu sehen, auf der alle Kunden, IFs und das Depot eingezeichnet sind. Das Depot ist in Quadrant $L13$. Die IFs befinden sich in den Quadranten $M29$, $F36$ und $J40$.

In der Tabelle 6.1 sind die zwölf Testszenarien und ihre Kenngrößen gegeben. Jedes Problem hat eine gewisse Anzahl von Kunden, welche über einen Planungszeitraum ($Tage$) besucht werden sollen. Je mehr Kunden zu besuchen sind und je kürzer der Planungszeitraum ist, desto mehr Fahrzeuge werden benötigt. Da es sich um eine mittelgroße Stadt handelt, sind zwei Fahrzeuge ausreichend. Die Fahrzeuge müssen, wenn sie voll sind, zu einer IF fahren. Es muss mindestens eine IF geben, es können, wie in einigen Szenarien, aber auch mehrere vorhanden sein.

Die Komplexität eines Szenarios ist nicht durch einen Parameter und auch nicht durch die Kenngrößen allein beschrieben, sondern ergibt sich aus der Kombination der Kenngrößen und der gewählten Parameter (siehe Tabelle 6.2). *Zeitfenster* komplizieren es, gültige Routen zu finden. Bei den Szenarien mit Zeitfenstern existieren 49 Kunden mit Zeitfenster. Diese Kunden müssen zwischen 06.30 Uhr und 12.00 Uhr besucht werden. Der *Zuordnungsmodus* bestimmt, wie der Abfall auf die Entladestationen verteilt werden soll. Die Wahl des Modus ist eine strategische Entscheidung, deren Folge auf die Routen nur schwer abschätzbar ist. Szenarien 07-09

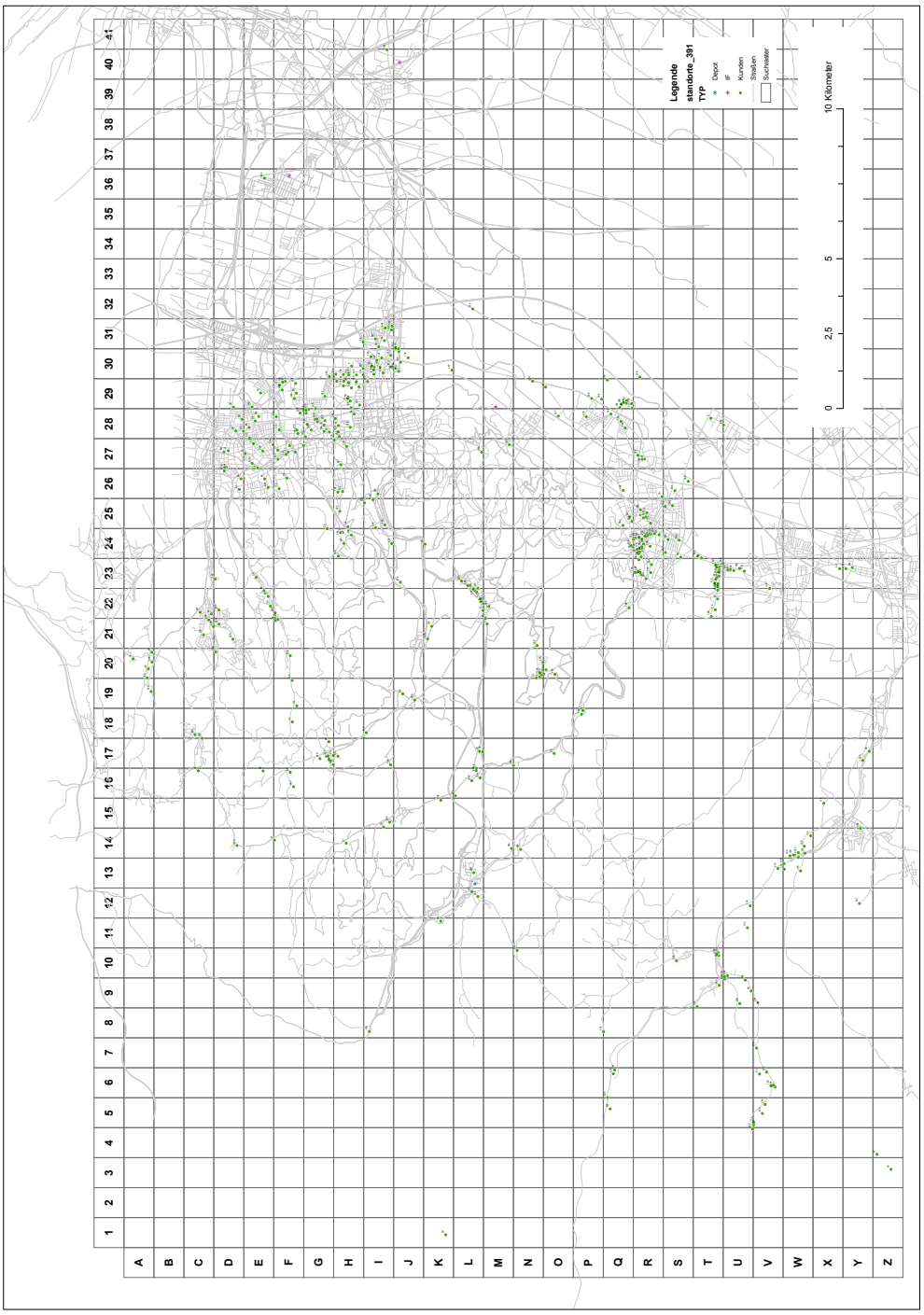



**Abbildung 6.1:** Geographische Darstellung der Testregion

und 10-11 haben dieselben Kenngrößen, jedoch unterschiedliche Parameter. Die Tests liefern daher auch einen Vergleich der Auswirkungen der verschiedenen Parameter.

Eine genauere Beschreibung der Kenngrößen und der Parameter ist in Kapitel 2 zu finden.

| Szenario | Kunden | Tage | Fahrzeuge | IFs |
|---|---|---|---|---|
| 01 | 92 | 1 | 1 | 2 |
| 02 | 184 | 5 | 1 | 1 |
| 03 | 184 | 5 | 1 | 1 |
| 04 | 387 | 5 | 2 | 3 |
| 05 | 387 | 5 | 2 | 3 |
| 06 | 387 | 5 | 2 | 3 |
| 07 | 387 | 5 | 2 | 3 |
| 08 | 387 | 5 | 2 | 3 |
| 09 | 387 | 5 | 2 | 3 |
| 10 | 387 | 5 | 2 | 3 |
| 11 | 387 | 5 | 2 | 3 |
| 12 | 387 | 5 | 2 | 3 |

**Tabelle 6.1:** Kenngrößen der Testinstanzen

| Szenario | Zeitfenster | IF Zuordnungsmodus |
|---|---|---|
| 01 | Nein | Frei |
| 02 | Nein | Frei |
| 03 | Nein | Frei |
| 04 | Nein | Frei |
| 05 | Nein | Frei |
| 06 | Nein | Frei |
| 07 | Nein | Frei |
| 08 | Nein | Prozentmäßig |
| 09 | Nein | Gemischt |
| 10 | Ja | Frei |
| 11 | Ja | Prozentmäßig |
| 12 | Ja | Gemischt |

**Tabelle 6.2:** Parameter der Testinstanzen

Bei Szenario 1 handelt es sich um ein degeneriertes PVRP. Es existiert nur ein Fahrzeug, und es wird auch nur für einen Tag geplant. Im Grunde handelt es sich daher um ein TSP. Szenario 2 und 3 sind auch Spezialfälle, da für die fünf Tage nur ein Fahrzeug zur Verfügung steht. Diese Instanzen können daher auch als PTSP bezeichnet werden. Zwischen dem Szenario 2 und 3 gibt es Unterschiede bei den Besuchsmustern, der Distanzmatrix und den Servicezeiten.

In der Praxis können immer wieder degenerierte PVRPs auftreten. Es ist daher essentiell, dass die Implementierung auch solche Fälle gut behandeln kann.

Szenario 4 und 5 unterscheiden sich nur in den Einsatzzeiten der Fahrzeuge (siehe Tabelle 6.3). Von Interesse ist der Vergleich zwischen den unterschiedlichen Einsatzzeiten. Es soll trotz der verringerten maximalen Einsatzzeit für die Woche ein Plan gefunden werden, der alle Bedingungen erfüllt. Szenario 6 und 7 bauen auf den vorherigen auf, besitzen jedoch für einige Kunden mehr Besuchsmuster. Hier ist interessant, ob diese größere Flexibilität genutzt werden kann und es zu besseren Routen kommt oder ob die Anzahl der hinzugekommenen Besuchsmuster zu gering ist.

Bei den Szenarien 7-12 handelt es sich um dieselben Testdaten. Sie unterscheiden sich nur in den Parametern. Das Ziel war, herauszufinden, wie sich die Lösungsqualität entwickelt, wenn zusätzliche Einschränkungen hinzugefügt werden.

| | Szenario 4 & 6 | | Szenario 5 & 7 | |
|---|---|---|---|---|
| Tag | max. Einsatzzeit | leer ins Depot | max. Einsatzzeit | leer ins Depot |
| Fahrzeug 1 | | | | |
| 1 | 0,0h | ja | 8,0h | nein |
| 2 | 8,5h | nein | 8,0h | nein |
| 3 | 8,5h | nein | 8,0h | nein |
| 4 | 8,5h | nein | 8,0h | nein |
| 5 | 8,5h | ja | 8,0h | ja |
| Fahrzeug 2 | | | | |
| 1 | 8,5h | nein | 5,0h | nein |
| 2 | 8,5h | nein | 8,0h | ja |
| 3 | 8,5h | nein | 9,0h | nein |
| 4 | 8,5h | ja | 9,0h | nein |
| 5 | 0,0h | ja | 9,0h | ja |

**Tabelle 6.3:** Einsatzzeiten für die Fahrzeuge von Szenario 4, 5, 6 und 7

## 6.2 Performance-Messung

Die Tests wurden auf einem Lenovo TS200v mit einem Intel Core i5 (3,2 GHz) Prozessor und 4 GB DDR3 Arbeitsspeicher (1333 MHz) durchgeführt. Dabei wurde nur ein Prozessorkern benutzt. Es kam Python 2.6.5 in Kombination mit Cython 0.14.1 und GCC 4.4.3 zum Einsatz.

Jedes Szenario wurde mit drei unterschiedlichen Verbesserungsverfahren getestet: Bei *VNS-SA* wird das VNS mit SA als Akzeptanzkriterium verwendet, wie es in den vorherigen Kapiteln beschrieben wurde. *VNS* benutzt für das VNS dieselben Parameter wie VNSSA, jedoch werden nur bessere Lösungen übernommen. Als drittes wird noch eine einfache lokale Suche verwendet, in dem Fall *2-Opt*. Die Verbesserungsverfahren bekommen ihre Startlösung von der Savings-Heuristik.

Für jedes Szenario und Lösungsverfahren wurden 30 Testläufe gestartet. Untersucht werden sollen zwei Qualitätskriterien:

- Die *Laufzeit* unter welcher im Folgenden die Zeit vom Starten des Programms bis zur abgeschlossenen Optimierung verstanden wird. Angegeben ist die Zeit in Minuten.

- Neben der Laufzeit sind vor allem die zu fahrenden Touren von großem Interesse. Unterschiede werden durch die benötigte Zeit pro Tourenplan quantifiziert. Es werden die Fahrzeiten zwischen den Kunden und IFs addiert und die Servicezeiten an jedem Punkt hinzugerechnet. Die Summe dieser Zeiten wird im Folgenden als *Einsatzzeit* bezeichnet und gibt die Güte einer Lösung an. Je kürzer die Einsatzzeiten einer Lösung desto besser.

Das erste Qualitätskriterium, welches untersucht werden soll, ist die Einsatzzeit. In der Tabelle 6.4 finden sich die Mittelwerte und Standardabweichungen für die Einsatzzeiten der Szenarien. Abbildung 6.2 stellt die Mittelwerte graphisch dar. In die Berechnung der Werte wurden nur die gültigen Lösungen miteinbezogen. Die Tabelle 6.4 weist die Anzahl der gefundenen gültigen Lösungen aus.

Am deutlichsten fallen die Einsatzzeiten der Szenarien 1, 2 und 3 auf. Diese Szenarien haben weniger Kunden zu besuchen, jedoch ähnliche Distanzen zwischen den Kunden und ähnliche Servicezeiten wie die restlichen Szenarien. Dadurch ergeben sich für diese Szenarien geringere Einsatzzeiten.

Der Vergleich der Szenarien 4-7 bringt ein unerwartetes Ergebnis. Obwohl die Szenarien 5 und 7 über eine größere maximale Einsatzzeit verfügen, sind die Ergebnisse schlechter als die von Szenario 4 und 6. Dafür konnte in jedem Testdurchlauf eine gültige Lösung gefunden werden. Für Szenario 6 kann ein besseres Ergebnis gefunden werden als für Szenario 4. Die Lösungen für Szenario 5 sind jedoch besser als die für Szenario 7. Hier wäre zu erwarten gewesen, dass auch Szenario 7 durch die zusätzlichen Besuchsmuster bessere Lösungen findet. Die gestiegene Komplexität, die durch die zusätzlichen Besuchsmuster entsteht, kann von der Implementierung nicht immer bewältigt werden. Bei der Erstellung von weiteren Szenarien in der Praxis sollte daher den Besuchsmustern die notwendige Aufmerksamkeit geschenkt werden.

Die Betrachtung der Lösungen für die Szenarien 7-9 zeigt, dass eine freie Zuteilung der IFs für die Einsatzzeiten am besten ist. Dass dieselben Lösungen für die volumsabhängige und gemischte Zuteilung gefunden wurden, lässt den Schluss zu, dass sich die fix zugeteilten IFs gut in den Plan eingliedern lassen. Aufgrund der geringen Anzahl der fix zugeteilten IFs (13) und der passenden Zuordnung kommt es in diesem Szenario zu keiner erhöhten Einsatzzeit. Es ist jedoch anzunehmen, dass sich die Einsatzzeiten erhöhen können, wenn ein größerer Anteil der Kunden eine fix zugeteilte IF hat. Dieselben Beobachtungen gelten auch für die Szenarien 10-12.

Interessant ist auch der Vergleich von Szenario 7 und 10. Szenario 10 besitzt Zeitfenster für 49 Kunden. Ansonsten unterscheiden sich die Szenarien nicht. Trotz der zusätzlichen Komplexität durch Zeitfenster unterscheiden sich die Lösungen für das Szenario 10 nicht um mehr als zehn Minuten von den Lösungen für Szenario 7.

Betrachtet man die Lösungen des VNS und VNS-SA, so stellt man fest, dass bei allen Szenarien mit freier IF-Zuteilung das VNS-SA leicht besser abschneidet. Bei den Szenarien 8, 9, 11 und 12 findet das VNS-SA zwar mehr gültige Lösungen, jedoch nicht immer in derselben Qualität wie das VNS.

2-Opt liefert nur für das erste Szenario eine gültige Lösung. Da es im 2-Opt nicht vorgesehen ist, dass die Besuchsmuster verändert werden, stellen alle Szenarien bei denen sich der Planungszeitraum über mehrere Tage erstreckt ein Problem dar.

| Szenario | 2-Opt | | | VNS | | | VNS-SA | | |
|---|---|---|---|---|---|---|---|---|---|
| | Mittel | $\sigma$ | gültig | Mittel | $\sigma$ | gültig | Mittel | $\sigma$ | gültig |
| 1 | 57 | 0 | 30 | **33** | 0 | 30 | **33** | 0 | 30 |
| 2 | – | – | 0 | 861 | 18 | 30 | **852** | 12 | 30 |
| 3 | – | – | 0 | 980 | 19 | 30 | **974** | 13 | 30 |
| 4 | – | – | 0 | 1883 | 39 | 29 | **1851** | 32 | 29 |
| 5 | – | – | 0 | 1939 | 40 | 30 | **1931** | 29 | 30 |
| 6 | – | – | 0 | 1876 | 85 | 29 | **1845** | 38 | 29 |
| 7 | – | – | 0 | 1960 | 42 | 30 | **1936** | 56 | 30 |
| 8 | – | – | 0 | **2231** | 157 | 25 | 2297 | 428 | 28 |
| 9 | – | – | 0 | **2231** | 157 | 25 | 2289 | 422 | 28 |
| 10 | – | – | 0 | 1963 | 40 | 30 | **1949** | 47 | 30 |
| 11 | – | – | 0 | **2249** | 151 | 25 | 2600 | 1360 | 26 |
| 12 | – | – | 0 | **2249** | 151 | 25 | 2600 | 1360 | 26 |

**Tabelle 6.4:** Einsatzzeiten für 2-Opt, VNS und VNS-SA

Als nächstes soll die Laufzeit betrachtet werden. Alle Algorithmen beenden die Optimierung im Durchshnitt in weniger als 60 Minuten. Nur einige Testläufe der Szenarien 8 und 9 mit VNS und VNS-SA brauchen knapp über 3 Stunden. Alle Testläufe des 2-Opt-Algorithmus können in 0,17 Minuten abgeschlossen werden.

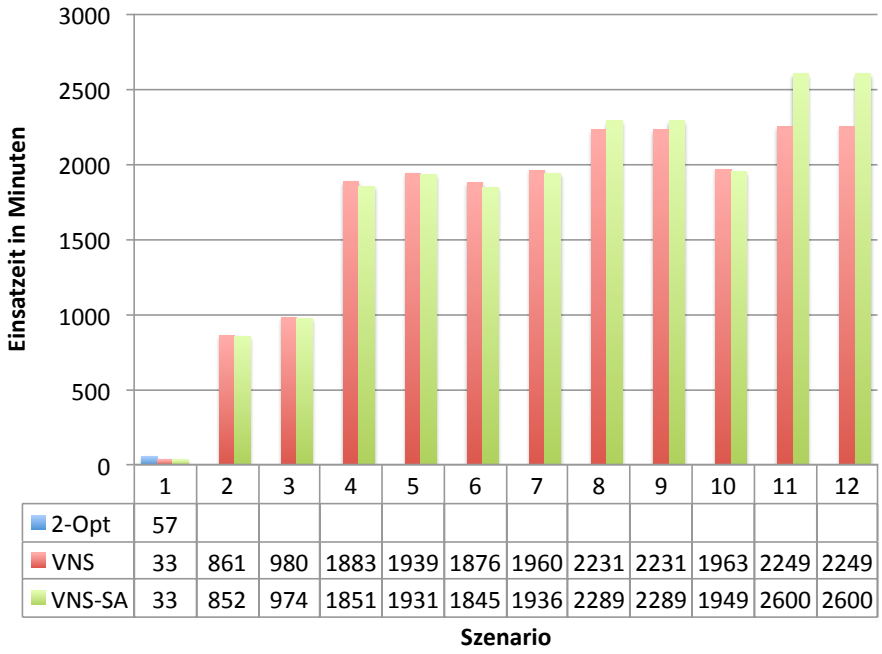
Bei den hier zu lösenden Problemen handelt es sich um strategische Aufgabenstellungen. In der Praxis werden Laufzeiten von mehreren Tagen noch als gut bewertet. Das bedeutet, dass selbst Laufzeiten von über 3 Stunden noch als sehr gut empfunden werden.

Abbildung 6.2 ist eine Gegenüberstellung der Einsatzzeiten für die getesteten Szenarien. Tabelle 6.5 zeigt die Mittelwerte und Standardabweichungen für alle Szenarien und Algorithmen.

Die Unterschiede in der Laufzeit zwischen Szenario 2 und 3 machen recht deutlich klar, dass allein die Kenngrößen keine Aussage über das Laufzeitverhalten geben können. Das bedeutet leider auch, dass keine genaue Vorhersage getroffen werden kann, wann die Optimierung fertig sein wird.

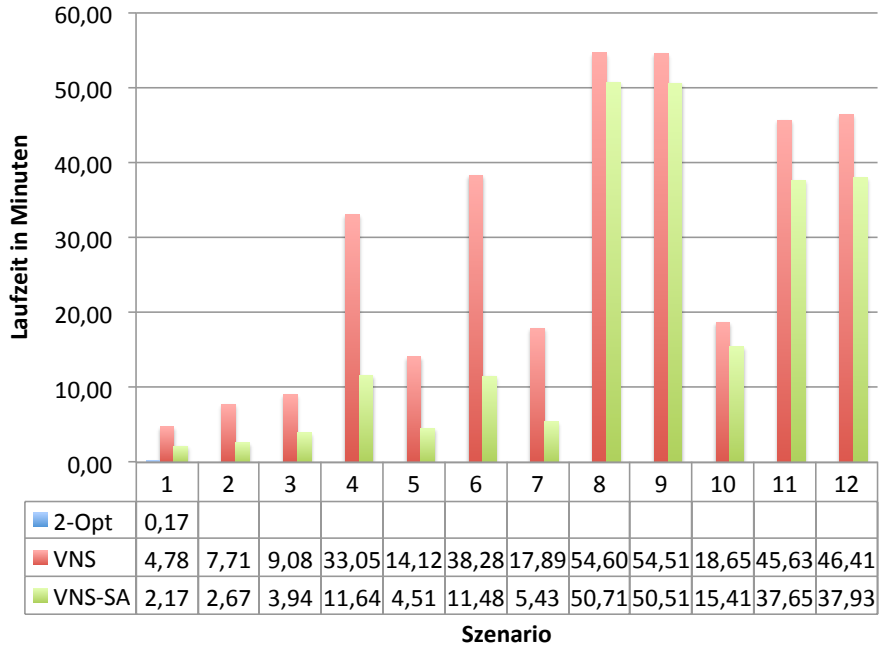
Einen gültigen Routenplan zu finden, wenn die Fahrzeuge nur 34 Stunden pro Woche zur Verfügung stehen, ist schwieriger, als wenn die Fahrzeuge 40 Stunden pro Woche eingesetzt werden können. Das schlägt sich auch in der Laufzeit nieder. Die Laufzeit von Szenario 4 ist um einiges länger als die von Szenario 5 bei VNS und VNS-SA.

Dieselbe Schlussfolgerung lässt sich aus den Laufzeiten der restlichen Szenarien ziehen. Die Wahl der Parameter verändert die Komplexität des Problems und führt dadurch zu einer längeren Laufzeit. Szenario 7 mit freier IF-Zuordnung und ohne Zeitfenster weist die geringste Komplexität und Laufzeit auf. Sind Zeitfenster zu beachten (Szenario 10), erhöht sich die Laufzeit. Noch länger dauert es, wenn eine alternative IF-Zuordnung gewählt wird. Die Kombination von alter-




| | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 2-Opt | 57 | | | | | | | | | | | |
| VNS | 33 | 861 | 980 | 1883 | 1939 | 1876 | 1960 | 2231 | 2231 | 1963 | 2249 | 2249 |
| VNS-SA | 33 | 852 | 974 | 1851 | 1931 | 1845 | 1936 | 2289 | 2289 | 1949 | 2600 | 2600 |

**Abbildung 6.2:** Einsatzzeiten pro Szenario in Minuten




| | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 2-Opt | 0,17 | | | | | | | | | | | |
| VNS | 4,78 | 7,71 | 9,08 | 33,05 | 14,12 | 38,28 | 17,89 | 54,60 | 54,51 | 18,65 | 45,63 | 46,41 |
| VNS-SA | 2,17 | 2,67 | 3,94 | 11,64 | 4,51 | 11,48 | 5,43 | 50,71 | 50,51 | 15,41 | 37,65 | 37,93 |

**Abbildung 6.3:** Laufzeiten pro Szenario in Minuten

| Szenario | 2-Opt | | | VNS | | | VNS-SA | | |
|---|---|---|---|---|---|---|---|---|---|
| | Mittel | $\sigma$ | gültig | Mittel | $\sigma$ | gültig | Mittel | $\sigma$ | gültig |
| 1 | **0,17** | 0 | 30 | 4,78 | 1,13 | 30 | 2,16 | 0,19 | 30 |
| 2 | – | – | 0 | 7,71 | 1,90 | 30 | **2,67** | 0,56 | 30 |
| 3 | – | – | 0 | 9,08 | 2,56 | 30 | **3,93** | 0,53 | 30 |
| 4 | – | – | 0 | 33,05 | 14,81 | 29 | **11,64** | 2,35 | 29 |
| 5 | – | – | 0 | 14,12 | 4,86 | 30 | **4,51** | 0,69 | 30 |
| 6 | – | – | 0 | 38,28 | 15,95 | 29 | **11,48** | 2,03 | 29 |
| 7 | – | – | 0 | 17,89 | 6,03 | 30 | **5,43** | 0,49 | 30 |
| 8 | – | – | 0 | 54,60 | 36,40 | 25 | **50,71** | 43,35 | 28 |
| 9 | – | – | 0 | 54,51 | 36,29 | 25 | **50,51** | 43,40 | 28 |
| 10 | – | – | 0 | 18,65 | 7,06 | 30 | **15,41** | 4,86 | 30 |
| 11 | – | – | 0 | 46,01 | 32,37 | 25 | **37,65** | 17,29 | 26 |
| 12 | – | – | 0 | 46,41 | 32,61 | 25 | **37,93** | 16,30 | 26 |

**Tabelle 6.5:** Laufzeiten für 2-Opt, VNS und VNS-SA

nativer IF-Zuordnung und Zeitfenster führt bei den Szenarien 11 und 12 zu kürzeren Laufzeiten, da für die Kunden mit Zeitfenstern weniger mögliche Besuchszeitpunkte zu untersuchen sind.

Vergleicht man die Laufzeiten von VNS mit VNS-SA, so fällt auf, dass VNS-SA in allen Szenarien bessere Laufzeiten liefert. VNS benötigt zwischen ein bis dreimal so lange wie VNS-SA. Erklären lässt sich der Unterschied durch die stochastische Natur von SA. Es werden auch Lösungen akzeptiert, die noch keine Verbesserung darstellen, aber in einigen Fällen schnell zu besseren Lösungen führen.

Es stehen leider keine historischen Daten zum Vergleich zur Verfügung. Daher kann nur der Disponent des Entsorgungsunternehmens eine ungefähre Einschätzung der Qualität der Lösung vornehmen. Eine Bewertung aufgrund von dessen langjähriger Erfahrung ergab, dass die mit dem Verfahren ermittelten Touren in den meisten Fällen kürzere Einsatzzeiten aufweisen als manuell geplante Touren.

Die positive Einschätzung der Lösungen durch den Disponenten bestätigt indirekt auch eine andere erfreuliche Eigenschaft der Implementierung: Das Verfahren liefert nicht nur bei typischen oder allgemeinen PVRPs-Instanzen gute Lösungen, sondern auch TSP- und PTSP-Probleme lassen sich damit lösen. Diese Generalität verspricht eine Anwendbarkeit bei sehr unterschiedlichen Fällen.

KAPITEL 7

# Conclusio und zukünftige Arbeiten

Die Abfallwirtschaft sucht nach effizienteren und besseren Planungswerkzeugen für die periodische Müllsammlung, um durch gezielteren Einsatz von Ressourcen die Kosten zu senken. Die vorliegende Arbeit beschreibt einen Lösungsansatz für ein Tourenplanungsproblem bei der Müllabholung von Sammelstellen. Das entwickelte Lösungsverfahren kann einen Planer unterstützen und ein erweitertes PVRP lösen. Dazu muss das Lösungsverfahren entscheiden, an welchen Tagen der Planungsperiode die Abfallbehälter einer Müllsammelinsel entleert werden und welches Fahrzeug diese Aufgabe übernehmen wird. Die Fahrzeuge besuchen mehrere Müllsammelstellen, und wenn ihre maximale Ladekapazität erreicht ist, fahren sie zu einer Entladestation. Dabei ist die Entladestation allerdings nicht immer frei wählbar, sondern sie wird durch einen Zuordnungsmodus bestimmt. Je nach Problem stehen als Zuordnungsmodus eine fixe Zuordnung für alle Müllsammelstellen oder eine fixe Zuordnung für nur einige Müllsammelstellen zur Verfügung. Weiters können eine anteilsmäßige Zuordnung nach Gesamtgewicht oder eine freie Zuordnung erfolgen. Schlussendlich kann auch eine Kombination aus den vier genannten Zuordnungsmodi vorgenommen werden. Als zusätzliche Erweiterung ist es möglich, dass Fahrzeuge am Ende des Tages beladen ins Depot zurückkehren und erst am nächsten Tag zu einer Entladestation fahren.

In der Fachliteratur wird dieses Problem als MDPVRP-IF bezeichnet, wobei die oben beschriebenen Erweiterungen darin nicht behandelt wurden. In dieser Arbeit wurde ein formales Modell für das MDPVRP-IF vorgestellt, welches auch zusätzliche Zuordnungsmodi für die IF berücksichtigt. Darüber hinaus kann abgebildet werden, wenn Fahrzeuge auch beladen ins Depot fahren.

Für das PVRP und Varianten davon wurden bereits viele Lösungsverfahren entwickelt - keines jedoch, das die vielen Nebenbedingungen des hier zu lösenden Problems berücksichtigt. Daher wurde als Lösungsverfahren ein VNS gewählt. Das VNS zeichnet sich durch seine große Flexibilität aus und liefert trotzdem gute Lösungen. Implementiert wurde das VNS in Python in Kombination mit Cython. Neben der Komponente mit dem Lösungsverfahren wurde ein Server entwickelt, sodass mehrere Probleme auf einem Rechner parallel gelöst werden können und die Hardware besser ausgelastet wird.

Getestet wurde das Lösungsverfahren mit realen Daten eines österreichischen Entsorgungsdienstleisters. Es wurden fünf unterschiedliche Testinstanzen zur Verfügung gestellt. Allerdings konnten keine historischen Vergleichswerte bereitgestellt werden. Die Lösungen des Verfahrens wurden von einem Disponenten des Entsorgungsunternehmens, das die Daten einbrachte, als sehr gut bezeichnet, und es gibt Überlegungen, die nächste Planung mit dem Lösungsverfahren durchzuführen.

Auch wenn das Lösungsverfahren bei dem vorgestellten Problem gute Lösungen liefert, gibt es in der Abfallwirtschaft noch viele Probleme und Spezialfälle, die nicht abgedeckt werden, z.B. die Hausmüllsammlung, bei der Straßenzüge abgefahren werden. Bei der Routenfindung muss unter anderem beachtet werden, dass LKWs nicht an jeder Kreuzung wenden können. Auch der Verkehr spielt hier eine viel wichtigere Rolle als bei der Entleerung von Müllsammelstellen. Denn die Entleerung aller Mistkübel entlang einer Straße braucht länger als die Entleerung einer Müllsammelinsel. So müssen die Müllsammelfahrzeuge, wenn sich Staus bilden, eine Runde um den Block fahren, um den Verkehr ganzer Stadtteile nicht komplett lahmzulegen, selbst wenn sie mit diesem Straßenabschnitt noch nicht fertig sind. Auch die Sicherheit des Personals ist ganz wichtig. Sind Straßen in beide Richtungen befahrbar, ist das Holen der Abfallcontainer von beiden Straßenseiten gefährlich. Stark befahrene Straßen sind daher in beide Richtungen abzufahren.

Die aktuell im Einatz befindlichen Müllsammelfahrzeuge können nicht mehrere Abfallsorten gleichzeitig einsammeln. Nur nach einer Entleerung und Reinigung kann eine andere Abfallsorte gesammelt werden. Derzeit wird immer nur für eine Abfallart optimiert. Besser wäre es jedoch, wenn bekannt wäre, welche unterschiedlichen Müllsorten abgeholt werden müssen und wie lange der Wechsel zwischen zwei Abfallarten benötigt. Ist zum Beispiel auf Grund der Besuchsmuster an einem Tag kein Restmüll abzuholen, könnte das Optimierungsverfahren am Ende des Vortags leer und gereinigt ins Depot fahren, um am nächsten Tag Papier einsammeln. Es ist zu untersuchen, ob sich dadurch die Fahrzeuge noch effizienter einsetzen lassen und eine weitere Kostenreduzierung erzielt werden kann.

Derzeit wird bei der Planung davon ausgegangen, dass die Müllcontainer immer voll sind. Bei der Abholung weisen die Container aber unterschiedliche Füllstände auf. Die LKWs fahren daher oft zu einer Entleerungsstation, obwohl sie noch nicht voll sind. Auch der Fahrer weiß beim Fahren der Tour nicht, wann und ob die maximale Ladekapazität erreicht ist. In Zukunft werden Fahrzeuge aber anzeigen können, wie voll sie sind. Eine dynamische Optimierung kann darauf eingehen und das Fahrzeug erst zu einer Entladestation schicken, wenn es notwendig ist und sich auszahlt. Doch auch Müllcontainer mit Füllstandsmessung werden zum Teil schon eingesetzt. Es müsste dann nur zu jenen Müllsammelstellen gefahren werden, welche auch einen gewissen Füllstand erreicht haben. Eine bedarfsgerechte Entleerung von Abfallbehältern bringt jedoch nicht nur eine Kostenersparnis und eine Schonung der Umwelt, sondern auch eine Verringerung des Lärms.

Eine dynamische Optimierung hätte freilich noch weitere Vorteile: Firmen, die kurzfristig eine Entleerung von Müllcontaineren benötigen, können besser in die Touren integriert werden. Aber auch auf Verkehrsbedingungen, welche gewisse Orte vorübergehend unzugänglich machen, kann besser reagiert werden. Dazu ist es allerdings notwendig, dass die LKWs mit Ortungsfunktionalität ausgestattet sind und das Verfahren schneller arbeitet.

Vor allem das Aufkommen von Multicore-Prozessoren und *General-Purpose Computing on Graphics Processing Units (GPGPU)* stellen hier neue Möglichkeiten dar, um die Berechnung zu beschleunigen. Doch nicht nur die Berechnungsgeschwindigkeit muss gesteigert werden, sondern auch die Lösungsverfahren müssen angepasst werden, um alle aktuellen Daten einbeziehen zu können.

# Literaturverzeichnis


[1] T. Grünert and S. Irnich. *Optimierung im Transport, Band 2: Wege und Touren*. Shaker Verlag, 2005.

[2] P. Hansen, N. Mladenović, and J. A. Moreno Pérez. Variable neighbourhood search: methods and applications. *Annals of Operations Research*, 175(1):367–407, 2010.

[3] V. C. Hemmelmayr, K. F. Doerner, and R. F. Hartl. A variable neighborhood search heuristic for periodic routing problems. *European Journal of Operational Research*, 195(3):791–802, 2009.

[4] K. Helsgaun. An effective implementation of the lin-kernighan traveling salesman heuristic. *European Journal of Operational Research*, 126(1):106–130, 2000.

[5] J. Puchinger. Combining metaheuristics and integer programming for solving cutting and packing problems. *PhD thesis, Vienna University of Technology, Institute of Computer Graphics and Algorithms*, supervised by G. R. Raidl and U. Pferschy, 2006.

[6] P. M. Francis, K. R. Smilowitz, and M. Tzur. The period vehicle routing problem and its extensions. *The Vehicle Routing Problem: Latest Advances and New Challenges*, 43:73–102, 2008.

[7] R. Russell and W. Igo. Assignment routing problem. *Networks*, 9(1):1–17, 1979.

[8] V. C. Hemmelmayr, K. F. Doerner, R. F. Hartl, and M. W. P. Savelsbergh. Delivery strategies for blood products supplies. *OR Spectrum*, 31(4):707–725, 2009.

[9] E. Angelelli and M. G. Speranza. The periodic vehicle routing problem with intermediate facilities. *European Journal of Operational Research*, 137(2):233–247, 2002.

[10] M. Mourgaya and F. Vanderbeck. Column generation based heuristic for tactical planning in multi-period vehicle routing. *European Journal of Operational Research*, 183(3):1028–1041, 2007.

[11] P. M. Francis, K. R. Smilowitz, and M. Tzur. The period vehicle routing problem with service choice. *Transportation Science*, 40(4):439–454, 2006.

[12] M. Polacek, R. F. Hartl, K. F. Doerner, and M. Reimann. A variable neighborhood search for the multi depot vehicle routing problem with time windows. *J Heuristics*, 10(6):613–627, 2004.

[13] J.-F. Cordeau, M. Gendreau, and G. Laporte. A tabu search heuristic for periodic and multi-depot vehicle routing problems. *Networks*, 30(2):105–119, Jan 1997.

[14] B. I. Kim, S. Kim, and S. Sahoo. Waste collection vehicle routing problem with time windows. *Computers & Operations Research*, 33(12):3624–3642, Jan 2006.

[15] G. Laporte, M. Gendreau, J.-Y. Potvin, and F. Semet. Classical and modern heuristics for the vehicle routing problem. *International Transactions in Operational Research*, 7(4-5):285–300, 2000.

[16] N. Mladenović and P. Hansen. Variable neighborhood search. *Computers & Operations Research*, 24(11):1097–1100, 1997.

[17] C. Blum and A. Roli. Metaheuristics in combinatorial optimization: Overview and conceptual comparison. *ACM Computing Surveys (CSUR)*, 35(3):268–308, 2003.