Hochschule für Angewandte Wissenschaften Hamburg
Hamburg University of Applied Sciences

Diplomarbeit

Rudolf Held

Konzeption und Umsetzung eines Verfahrens zum interaktiven und dynamischen Flottenmanagement

Fakultät Technik und Informatik
Department Informatik

Faculty of Engineering and Computer Science
Department of Computer Science

Rudolf Held

Konzeption und Umsetzung eines Verfahrens zum interaktiven und dynamischen Flottenmanagement

Diplomarbeit eingereicht im Rahmen der Diplomprüfung
im Studiengang Softwaretechnik
am Department Informatik
der Fakultät Technik und Informatik
der Hochschule für Angewandte Wissenschaften Hamburg

Betreuender Prüfer : Prof. Dr. rer. nat. Michael Neitzke
Zweitgutachter : Prof. Dr. rer. nat. Christoph Klauck

Abgegeben am 28. August 2008

Rudolf Held

Thema der Diplomarbeit

Konzeption und Umsetzung eines Verfahrens zum interaktiven und dynamischen Flottenmanagement

Stichworte

Dynamisches Flottenmanagement, logistische Systeme, Tourenplanung, Routenplanung, Tourenoptimierung, Routenoptimierung, Vehicle Routing Problem, interaktive Optimierung, Real–Time Routing

Kurzzusammenfassung

In dieser Arbeit werden die aus dem Vehicle Routing Problem Bereich bekannten Algorithmen auf ihre Verwendbarkeit in Systemen zur interaktiven und dynamischen Planung und Optimierung von Touren untersucht. Es wird das Design eines Systems zum interaktiven und dynamischen Management von Fahrzeugflotten entworfen und seine Anwendbarkeit anhand der Implementierung eines Prototyps verifiziert. Konzeption und Realisierung finden dabei unter besonderer Berücksichtigung der umgebungsspezifischen Randbedingungen im Werttransport–Bereich statt.

Rudolf Held

Title of the paper

Conception, Design and Prototypical Realization of a System for Interactive and Dynamic Fleet Management

Keywords

Dynamic fleet management, logistic systems, tour planning, route planning, tour optimization, route optimization, vehicle routing problem, interactive optimization, real–time routing

Abstract

In this thesis the suitability of algorithms known in Vehicle Routing Problem field for application in systems for interactive and dynamic route–planning and –optimization is examined. A system for interactive and dynamic fleet management is designed and it's applicability verified by the implementation of a prototype. Design and realization are done in view of the special requirements for logistic systems in the area of cash and valuable goods transportation.

Danksagung

Meinem Betreuer Prof. Dr. Michael Neitzke möchte ich für das spannende Thema meiner Arbeit danken.

Meinem Betreuer Prof. Dr. Christoph Klauck möchte ich nicht nur für die Betreuung während dieser Arbeit, sondern auch für die Betreuung während meines Studiums and der HAW danken. Sein Engagement war für mich eine große Hilfe auf meinem Weg zum Diplom.

Meiner Mutter Gertraud Held und meinem Großvater Philipp Glogger möchte ich für ihre Unterstützung in der Endphase des Studiums danken.

Bei meinen Freunden und Freundinnen, besonders Kolja und Oda, Malte und Verena, möchte ich mich für ihr Verständnis, mich in der letzten Zeit des Studiums kaum noch gesehen zu haben, bedanken.

Danke, Annette, dass für mich da warst und diese Zeit mit durchgestanden hast. Deine Hilfe und dein Verständnis waren mit entscheidend für den Erfolg dieser Arbeit.

Inhaltsverzeichnis

Tabellenverzeichnis

Abbildungsverzeichnis

1 Einführung

Das dynamische Management von Fahrzeugflotten im Logistikbereich gewinnt zunehmend an Bedeutung. Mehrere Faktoren wirken verstärkend auf diesen Trend: die Entwicklung von hochentwickelten Kommunikationstechnologien und telematischem Equipment, die Verfügbarkeit kommerziell nutzbarer geographischer Informationssysteme sowie die Fortschritte in der Wissenschaft bei der Entwicklung von Modellen der im Logistikbereich auftretenden Probleme sowie bei den Lösungsansätzen für die entsprechenden Problemstellungen. Weitere Faktoren sind der zunehmende Wettbewerbsdruck bei steigenden Anforderungen an logistische Systeme und die Dringlichkeit einer effizienteren Nutzung von Energie im Transportwesen zur Schonung von Energieressourcen und Umwelt. Ein dynamisches Flottenmanagement ermöglicht unter Ausnutzung bestehender Technologien und Lösungsansätze zeitnahe Reaktion auf sich dynamisch während des operativen Einsatzes von Transportfahrzeugen verändernde Bedingungen.

Die Anregung zu dieser Arbeit stammt von der Firma IT–Kompetenz.

1.1 Motivation

Logistische Systeme sind die Grundlage für die in unserer Gesellschaft notwendige Mobilität von Gütern und Personen. Eines der Hauptprobleme bei der Planung im logistischen Bereich ist die Optimierung von Transportrouten. Diese sind meist in Hinsicht auf die Anzahl der einzusetzenden Fahrzeuge sowie die Längen und zeitliche Dauer der Transportrouten zu optimieren. Abhängig vom Umfeld ist auch die Fähigkeit zur schnellen Reaktion auf spontane Anfragen und zur termingerechten Abwicklung innerhalb enger zeitlicher Toleranzen gefordert. Die Anforderungen an Systeme zur Planung, Kontrolle und Steuerung von logistischen Prozessen sind dabei in den letzten Jahren enorm gestiegen. Die Menge an täglich beförderten Gütern und Personen ist immens, mit steigender Tendenz. So schätzt das Bundesamt für Verkehr, Bau und Stadtentwicklung die Zunahme der Güterverkehrsleistung im Binnen-Strassengüterverkehr in Deutschland von 237 Mrd. Tonnenkilometer im Jahr 2005 auf 315 Mrd. Tonnenkilometer im Jahr 2020 [vgl. Ickert u. a., 2007, S.123]. Die mit der Optimierung von Transportrouten verbundene Einsparung an Ressourcen verringert nicht nur Transportkosten, sondern auch die durch den Verkehr verursachten Umweltbelastungen.

Bei letzterem Punkt wird auch zunehmend der Gesetzgeber aktiv. Da sich Deutschland im Rahmen des Kyoto–Protokolls dazu verpflichtet hat, seine Treibhausgasemissionen bis 2012 um 21 Prozent gegenüber 1990 zu reduzieren, versucht die Bundesregierung den in den letzten Jahren anhaltenden Trend des Emisionsrückgangs von Schadstoffen weiter zu verstärken [vgl. Schafhausen u. a., 2005, S. 4]. Als erster von vier Handlungsschwerpunkten für den Sektor Verkehr gilt dabei der Punkt "Anreizmechanismen zur Verminderung der Transportintensität und zur Steigerung der Energieeffizienz des Verkehrssektors" [Schafhausen u. a., 2005, S. 46].

Eine *a priori* erstellte Routenplanung, die rein statisch ist und sich während der Ausführung nicht ändert, bringt einige Nachteile mit sich. Veränderte Umweltbedingungen wie umschlagende Wetterverhältnisse oder eine sich ändernde Verkehrssituation können die Ausführbarkeit eines statischen Plans unmöglich machen. Eine Berücksichtigung spontaner Anfragen, die während der Ausführung auftreten, ist nicht möglich. Andererseits eröffnet die Verfügbarkeit von Positionsermittlungs– und Kommunikationssystemen wie dem Global Positioning System (GPS) und dem General Packet Radio System (GPRS) sowie die steigende Rechenkapazität mobiler Geräte wie Personal Digital Assistants (PDA) und Smartphones neue Möglichkeiten zur Informationsgewinnung und Verarbeitung. Dies wiederum bietet neue Möglichkeiten für Entwicklungen im Bereich von Systemen zur Entscheidungs–Unterstützung (Decision Support Systems). ICHOUA bemerkt, dass die Verfügbarkeit neuer Kommunikationtechnologien und die Notwendigkeit besserer Decision Support Systems in den letzten beiden Jahrzehnten zu einem verstärkten Interesse an Systemen zum dynamischen Flottenmanagement geführt haben [vgl. Ichoua u. a., 2000, S.427].

Ein weiterer Gegenstand aktueller wissenschaftlicher Forschung ist die Entwicklung interaktiver Systeme zur Routenplanung. Bei interaktiven Systemen kann ein Dispatcher in den Optimierungsprozess eingreifen und diesen steuern, indem er Parameter des Optimierungsalgorithmus oder die Zuordnung von Kunden zu bestimmten Routen ändert.

Logistikfirmen profitieren von einer dynamischen Optimierung der Fahrtrouten ihrer Fahrzeugflotten in mehrfacher Hinsicht: die Einsparung von Kosten erhöht die Wettbewerbsfähigkeit. Die durch Optimierung der Fahrtrouten erhöhte Effizienz erhöht das Reaktionsvermögen, mit dem auf veränderte Umweltbedingungen wie Wetter oder Verkehrssituation sowie auf spontane Kundenanfragen reagiert werden kann. Insgesamt kann damit auch bei hohen Anforderungen ein Kunden-Service auf hohem Niveau gewährleistet werden. Die Verringerung der durch die operative Tätigkeit verursachten negativen Auswirkungen auf die Umwelt sorgt für ein besseres Image der Firma.

1.2 Umfeld

Das Umfeld dieser Arbeit ist im Bereich der Werttransporte angesiedelt. Die hier dargestellten Informationen über das Umfeld wurden der Internetpräsenz der Firma Brink's Deutschland GmbH (http://www.brinks.de) [1] entnommen. Werttransportfirmen wie z. B. die Firma Brink's Deutschland GmbH bieten den gesicherten Transport von Geld– und anderen Werten in gesicherten Spezialfahrzeugen an. Das Spektrum der Kunden, die diese Dienstleistungen in Anspruch nehmen, reicht von Einzelhändlern über größere gewerbliche Kunden wie Supermarktketten bis hin zu Banken. Die Palette der Dienstleistungen ist dabei vielfältig. Während bei Einzelhändlern hauptsächlich die Tageseinnahmen abzuholen und zur nächsten Bank zu transportieren sind, lassen sich Supermärkte oft zusätzlich zur Abholung der Tageseinnahmen bei Bedarf mit Wechselgeld versorgen. Größere Kunden und Banken nehmen oft einen B2B–Service (Business to Business) in Anspruch, bei dem Valora (Werte) zwischen zwei Kunden oder zwei Filialen eines Kunden zu tauschen sind. Zusätzlich wird die Zählung, Verifizierung und Bündelung von Banknoten und Münzen angeboten. Ein weiteres Dienstleistungsangebot ist die Bestückung und technische Betreuung von Geldautomaten, nach Abkürzung der englischen Bezeichnung "Automatic Teller Machine" auch ATM genannt.

Aufgrund des Dienstleistungsangebots ergeben sich für die Transportfahrzeuge drei Arten von Transporten, die sich in der Art von Start– und Zielpunkt eines jeweiligen Transports von Valora unterscheiden: Transporte vom Depot aus zu einem Kunden, Transporte von einem Kunden aus zum Depot und Transporte von einem Kunden zu einem anderen Kunden bzw. von einer Filiale eines Kunden zur einer anderen Filiale des Kunden.

Die Art und Weise der Durchführung von Geldtransporten ist durch die Unfallverhütungsvorschrift der Wach– und Sicherheitsdienste, BGV C7, verbindlich festgelegt. Zur Abwicklung der operativen Tätigkeit unterhalten Werttransportfirmen Depots. Diese sind mit Sicherheitsschleusen für die Transportfahrzeuge auszustatten [vgl. Paulick, 2007, S.2, S.4]. Diese Schleusen sind von den Fahrzeugen beim Verlassen des Depots und bei der Anfahrt zum Depot zu passieren. Die Bearbeitungszeiten der Fahrzeuge innerhalb der Schleuse sind für das Verlassen des und die Anfahrt zum Depot verschieden.

Aufgrund der Bestimmungen des BGV C7 muss ein Geldtransportfahrzeug während einer Tour ständig von mindestens einer Person besetzt sein. Aus Sicherheitsgründen und zur Wahrung des Vier–Augen–Prinzips bei der Handhabung von Valora muss die Bestückung und technische Betreuung von Geldautomaten durch mindestens zwei Personen durchgeführt werden. Die Abholung bzw. Abgabe von Valora bei Kunden wie z. B. Einzelhändlern oder Banken, bei denen das Vier–Augen–Prinzip bei der Übernahme von Valora jeweils

[1] Links zu den einzelnen Informationen können nicht angegeben werden, da die Seite auf Frames basiert.

durch ein Besatzungsmitglied des Transportfahrzeugs und dem Kunden oder einer Vertrauensperson des Kunden gewahrt wird, kann durch ein Besatzungsmitglied des Transportfahrzeugs durchgeführt werden. Voraussetzung dafür sind spezielle Sicherungsmassnahmen wie besonders gesicherte Transportbehälter, die im Fall eines Überfalls die Valora durch Einfärbung wertlos machen [vgl. BGV:1997, 1997, S.8–10]. Somit ergeben sich zwei Arten von Aufträgen, die sich in der Mindestbesatzungsstärke eines Transportfahrzeugs unterscheiden: Aufträge, bei denen die Mindestbesatzungsstärke zwei Mann pro Transportfahrzeug beträgt sowie solche, bei denen die Mindestbesatzungsstärke drei Mann beträgt.

Für die Kapazität der Transportfahrzeuge besteht eine Obergrenze, die versicherungstechnisch begründet ist, d. h. zu jedem Zeitpunkt darf die in einem Transportfahrzeug enthaltene Menge an Valora einen bestimmten Betrag an Wert nicht überschreiten. Die Gewichts– bzw. Volumenkapazität des Fahrzeugs ist aus diesem Grund vernachlässigbar, da die Obergrenzen dieser Kapazitäten im normalen Einsatz nicht erreicht werden.

1.3 Randbedingungen der Problemstellung

Der Zeitraum, für den Touren geplant werden, ist ein Arbeitstag. Der Zeitraum, für den eine Planung stattfindet, wird im VRP–Bereich üblicherweise als *Planungshorizont* (engl. *Planning Horizon*) bezeichnet [vgl. u. a. Larsen u. a., 2007, S.27]. Ausgangs– und Endpunkt aller Touren ist das Depot des Werttransporteurs. Bei der Optimierung sind als Kostenfaktoren sowohl die Gesamtstrecke (Fahrzeugkosten) als auch die Gesamtdauer (Personalkosten) der Tour zu berücksichtigen. Aufträge zur Belieferung mit und Abholung von Werten, im nachfolgenden *Requests* genannt, können vor der *a priori* Planung für einen Planungshorizont bekannt sein oder während des Planungshorizonts auftreten (*spontane Requests*).

Es sind folgende Arten von Requests möglich: Auslieferung von Werten vom Depot aus zu einem Kunden (*Delivery*), Abholung von Werten bei einem Kunden und Auslieferung dieser Werte bei einem anderen Kunden bzw. bei einer anderen Filiale des Kunden (*Pickup and Delivery*), Abholung von Werten bei einem Kunden und Transport dieser Werte zum Depot (*Pickup*). Dabei bedingt die Art des Auftrags (z. B. Supermarkt oder ATM), die jeweils in einem Request enthalten ist, die minimale Besatzungsstärke von Fahrzeugen, denen der jeweilige Request zugeordnet werden kann. Requests sind nur den Touren zuzuweisen, deren Fahrzeug die für den jeweiligen Request erforderliche Besatzungsstärke aufweisen. Bei einem Request kann für Pickup und/oder Delivery jeweils ein Zeitfenster vergeben werden.

Ereignisse, die während des Planungshorizonts auftreten und das Optimierungsproblem beeinflussen oder verändern, werden im folgenden als *Events* bezeichnet. Beispiele für Events sind spontane Requests oder Statusmeldungen von Transportfahrzeugen. Die Behandlung

spontaner Requests soll möglich sein. Spontane Requests sind – soweit möglich – in eine bestehende Tour zu integrieren oder einem Ersatzfahrzeug zuzuweisen.

Das Depot enthält Schleusen, die die Transportfahrzeuge beim Verlassen des Depots bzw. bei der Rückkehr zum Depot passieren müssen. Pro Zeiteinheit kann in einer Schleuse nur eine bestimmte Anzahl von Fahrzeugen bearbeitet werden. Die Bearbeitungszeiten für Fahrzeuge in der Schleuse beim Verlassen des und der Rückkehr zum Depot sind jeweils unterschiedlich. Zurückkehrende Fahrzeuge sollen das Depot möglichst ohne Wartezeit befahren können.

Die Fahrtrouten sind anhand der geographischen Standorte und dem jeweiligen Bedarf der Kunden sowie anhand der Anzahl der zur Verfügung stehenden Fahrzeuge zu optimieren.

1.4 Zielsetzung

Ziel dieser Arbeit ist die Untersuchung bekannter existierender Lösungsansätze hinsichtlich ihrer Eignung als Basis zur Lösung der Problemstellung sowie die Erstellung eines Designs für ein Systems zur Optimierung von Fahrtrouten von Werttransportern unter Berücksichtigung dynamischer Eigenschaften des Problems wie z. B. spontane Kundenanfragen. Soweit möglich, soll eine Umsetzung des Designs in einen Prototypen erfolgen. Geographische Standorte werden in euklidischen Koordinaten angegeben.

Die Benutzerschnittstelle des Systems soll die Möglichkeit bieten, in geeigneter Weise interaktiv in den Optimierungsprozess eingreifen zu können. Die Frage, welche Interaktionsmöglichkeiten notwendig bzw. sinnvoll sind, ist durch eine entsprechende Anaylse zu klären.

Die Problematik der Schleusen des Depots und der minimalen Besatzungsstärke von Fahrzeugen zur Abwicklung von Requests stellt eine besondere Anforderung dar. Die Berücksichtigung dieser Kriterien bei der Optimierung ist ausschlaggebend für die Anwendbarkeit der Lösung. Es ist jedoch vor einer eingehenden Analyse der existierenden Lösungsansätze schwer absehbar, inwieweit innerhalb dieser Arbeit die Erarbeitung einer Lösung machbar ist, die auch hinsichtlich dieser Kriterien eine vollautomatische Optimierung bietet. Eine Lösung, die die Möglichkeit der Optimierung durch Kooperation eines menschlichen Planers (Dispatcher) und dem Optimierungssystem bietet, wird deswegen als ausreichend erachtet.

1.5 Aufbau der Arbeit

In diesem Kapitel werden die Motivation für die Arbeit sowie das Umfeld und die Zielsetzung beschrieben.

Im zweiten Kapitel werden die benötigten Grundlagen aus dem Bereich der Optimierung von Routing Problemen vorgestellt. Hierzu gehören die verschiedenen Arten von Routing Problemen, die Komplexität von Routing Problemen und Verfahren zur Optimierung von Routing Problemen.

Im dritten Kapitel erfolgt eine Analyse der Problemstellung sowie der Anforderungen an ein System zur Behandlung der Problemstellung. Die im zweiten Kapitel vorgestellten Optimierungs–Verfahren werden analysiert und deren Eignung zur Anwendung auf die Problemstellung wird untersucht.

Im vierten Kapitel wird das Design des Systems entwickelt. Dabei wird auf die Architektur des Gesamtsystems, die einzelnen Komponenten und ihr Zusammenspiel sowie auf das Design des Algorithmus eingegangen.

Die Realisierung des prototypischen Systems wird im fünften Kapitel behandelt. Dabei wird die Umsetzung der aus der Analyse stammenden Überlegungen sowie der im Kapitel Design vorgestellten Konzepte beschrieben.

Im sechsten Kapitel wird das Ergebnis der Arbeit mit der eingangs genannten Zielsetzung verglichen. Zum Schluss erfolgt ein Ausblick auf weitere mögliche Weiterentwicklungen.

2 Grundlagen

In diesem Kapitel werden die – meist aus dem Bereich des Operations Research stammenden – Grundlagen zum Thema Routenoptimierung vorgestellt. KLAUCK UND MASS beschreiben das Operations Research (OR) als das Wissenschaftsgebiet, welches sich mit der Optimierung von Arbeitsabläufen anhand quantifizierbarer Daten und im Sinne quantitativ formulierter Ziele befasst, wobei Hilfsmittel aus den verschiedensten Teilgebieten der Mathematik benutzt werden [vgl. Klauck und Maas, 1999, S.9]. Am Anfang dieses Kapitels werden das Basismodell zur Modellierung von Routing Problemen, das *Vehicle Routing Problem* (VRP) und einige erweiterte Varianten des VRP werden vorgestellt. Im Teilbereich des Operations Research, der sich mit Routing Problemen befasst, werden zum Testen von Lösungsverfahren Benchmark Probleme eingesetzt. Die bekanntesten Benchmarks werden hier ebenfalls dargestellt. Den breitesten Raum in diesem Kaptiel nehmen die Verfahren zur Lösung von Vehicle Routing Problemen ein. Ausgehend von den ab den 1960er Jahren entwickelten klassischen Verfahren für statische VRP werden exakte Verfahren und die ab den 1980er Jahren Bedeutung erlangenden Metaheuristiken für statische VRP behandelt. Im Anschluss daran wird auf die Lösungsansätze und Verfahren zur Behandlung von dynamisch–stochastischen VRP eingegangen. Gegen Ende dieses Kapitels werden Ansätze für interaktive Verfahren zur Lösung von VRP dargestellt.

Der Teilbereich des Operations Research, der sich mit der Optimierung von Vehicle Routing Problemen befasst, ist ausserordentlich weitläufig. Die Darstellung der in diesem Kapitel behandelten Themen kann deshalb nur einen ersten Einblick geben und erfolgt eng begrenzt soweit, wie es für die Aufgabenstellung dieser Arbeit nötig ist.

2.1 Das Vehicle Routing Problem

Im wissenschaftlichen Bereich wurde das Problem der Routenoptimierung 1959 von G.B. DANZIG UND J.H. RAMSER als *Truck Dispatching Problem* (DTP) eingeführt. Die Problemstellung besteht dabei in der Belieferung von Tankstellen von einem zentralen Depot aus durch Tanklastwagen. Dabei sind die Tankstellen den Tanklastwagen so zuzuordnen, dass die Summe der einzelnen Strecken der Tanklastwagen minimiert wird [vgl. Danzig und Ramser, 1959, S.1]. Das Probem wurde später unter dem Namen “Vehicle Routing Problem”

(VRP) bekannt. Das Problem wird durch einen vollständigen Graph dargestellt. Das Depot und die Kunden werden durch die Ecken des Graphen repräsentiert. Die Entfernungen zwischen den Kunden sowie zwischen Kunden und Depot sind durch die Gewichtsmatrix des Graphen definiert. Vom Vehicle Routing Problem bestehen etliche Unterarten, die jeweils ein oder mehrere Randbedingungen für die Optimierung beinhalten. Ein Beispiel ist das "Vehicle Routing Problem with Time Windows" (VRPTW), bei dem die Kunden jeweils innerhalb eines bestimmten Zeitfensters anzufahren sind.

2.1.1 Vehicle Routing Problem – Varianten

Der Autor identifizierte während der Literatur–Recherche zu dieser Arbeit ca. 25 VRP–Varianten und geht davon aus, dass noch einige weitere Varianten existieren. Eine Auflistung aller identifizierten VRP–Varianten in dieser Arbeit erscheint als wenig sinnvoll, da dies zum einen den Rahmen der Arbeit sprengen würde und zum anderen viele VRP–Varianten Problemstellungen modellieren, deren Randbedingungen für die Problemstellung dieser Arbeit nicht relevant sind. Die für diese Arbeit relevanten Varianten werden im folgenden vorgestellt.

2.1.1.1 Capacitated Vehicle Routing Problem (CVRP)

Bei der einfachsten Variante der Routing Probleme werden Kunden von einem Depot mit Gütern beliefert, wobei die Fahrzeugflotte als homogen angenommen wird und jedes Fahrzeug eine definierte Kapazität besitzt. [vgl. Coltorti und Rizzoli, 2007, S.3] Die Bedingung, dass die Routen der Fahrzeuge an einem zentralen Depot starten und enden, wird meist nicht angegeben.

2.1.1.2 Vehicle Routing Problem with Time Windows (VRPTW)

Das Vehicle Routing Problem With Time Windows (VRPTW) ist bei GAMBARDELLA U. A. als Erweiterung des Capacitated Vehicle Routing Problem (CVRP) definiert. Sowohl dem Depot als auch jedem Kunden ist ein Zeitfenster zugeordnet. Das Zeitfenster des Depots definiert die früheste Zeit, zu der Fahrzeuge vom Depot aus starten können sowie den spätesten Zeitpunkt für die Rückkehr zum Depot. Die Fahrzeuge müssen bei einem Kunden innerhalb des Zeitfensters des Kunden eintreffen [vgl. Gambardella u. a., 1999, S.3–4].

Bei COLTORTI UND RIZZOLI sind beim VRPTW nur Zeitfenster für Kunden angegeben [vgl. Coltorti und Rizzoli, 2007, S.3].

2.1.1.3 Pickup and Delivery Problem (PDP)

CORDEAU U. A. beschreiben das Pickup and Delivery Problem als Problem, bei dem eine Menge von jeweils paarweisen Kundenbeziehungen besteht. Dabei müssen Güter von einem Kunden zu einem anderen Kunden transportiert werden. Die Routen aller Fahrzeuge starten und enden an einem zentralen Depot. Das Ziel ist, die Menge von Routen zu berechnen, bei der die Gesamtkosten aller Routen unter Berücksichtigung der gegebenen Randbedingungen minimal wird. Die Autoren definieren das Problem als one–to–one Pickup and Delivery Problem und grenzen es explizit vom one–to–many Problem ab, bei dem eine Menge von Kunden von einem zentralen Depot aus mit Waren beliefert werden und Waren an das zentrale Depot abgeben. Das Problem wird ebenfalls von many–to–many Problemen abgegrenzt, bei denen die Kundenbeziehungen nicht mehr zwingend paarweise sein müssen [vgl. Cordeau u. a., Due 2008, S.1–2].

SAVELSBERGH UND SOL beschreiben das Pickup and Delivery Problem (PDP) als Problem, bei dem generell Güter von einem Kunden zu einem anderen Kunden transportiert wird, ohne den Zusatz one–to–many zu nennen [vgl. Savelsbergh und Sol, 1995, S.1]

Werden anstatt Gütern Menschen befördert, wird das Problem als Dial–a–Ride Problem (DARP) bezeichnet [vgl. Cordeau u. a., Due 2008; Savelsbergh und Sol, 1995, S.1].

Wie beim VRP existiert auch beim PDP eine Variante mit Zeitfenstern, die in der Literatur als Pickup and Delivery Problem With Tim Windows (PDPTW) bekannt ist. Beim PDPTW sind für jeden Request zwei Zeitfenster definiert, innerhalb deren die Abholung bzw. Abgabe der Waren zu erfolgen hat. Bei CORDEAU U. A. sind Zeitfenster grundsätzlicher Bestandteil eines PDP [vgl. Cordeau u. a., Due 2008, S.1]. SAVELSBERGH UND SOL definieren eine Erweiterung des PDP, das General Pickup and Delivery Problem (GPDP), um verschiedene Randbedingungen, die in realen Anwendungen auftreten, abzubilden. Neben Zeitfenstern für die Requests können zusätzlich auch Zeitfenster für Fahrzeuge vergeben werden, um z. B. Ruhezeiten für Fahrer abzubilden [vgl. Savelsbergh und Sol, 1995, S.1–10].

Ein sehr reichhaltiges Modell des PDPTW findet sich bei ROPKE UND PISINGER. Zusätzlich zu den Zeitfenstern für einen Request können hier auch Start– und Endzeit eines Fahrzeugs definiert werden. Für einen Pickup bzw. Delivery Punkt kann die Bearbeitungszeit zum Beladen bzw. Entladen definiert werden. Die Fahrzeuge müssen nicht zwingenderweise über gleiche Eigenschaften hinsichtlich der Eignung zum Transport bestimmter Güter aufweisen. Jedem Request kann eine Menge von Fahrzeugen zugeordnet werden, die in der Lage sind, den Request auszuführen. Alle Fahrzeuge haben einen definierten Start– und Endpunkt, der sowohl beim einzelnen Fahrzeug als auch bei den Fahrzeugen untereinander unterschiedlich sein kann. Das bedeutet, dass Fahrzeuge von verschiedenen Punkten aus starten können und Start– und Endpunkt eines Fahrzeugs unterschiedlich sein können. Dadurch werden

auch Eigenschaften des Multiple Depot Vehicle Routing Problem (MDVRP) modelliert. Explizit erwähnt wird die Möglichkeit, dass Fahrzeuge einen Pickup bzw. Delivery Punkt vor Beginn des entsprechenden Zeitfensters befahren können. In diesem Fall ist jedoch der Beginn der Zeitfensters abzuwarten, bevor die Beladung bzw.. Entladung stattfinden kann [vgl. Ropke und Pisinger, 2006a, S.1–2].

Bei allen in diesem Abschnitt genannten Autoren beinhaltet das Modell die Kapazität der Fahrzeuge.

2.1.1.4 Dynamic Vehicle Routing Problem (DVRP)

LARSEN U. A. beschreiben das Dynamic Vehicle Routing Problem (DVRP) als Erweiterung des klassischen VRP, bei der eine Teilmenge von Requests bekannt wird, nachdem die Ausführung des usrprünglichen Plans begonnen hat [vgl. Larsen u. a., 2007, S.20]. Die Dynamik des Problems hat weitreichende Konsequenzen, von denen die Autoren u. a. folgende nennen:

1. Während der Ausführung muss dem Dispatcher zu jedem Zeitpunkt die Position aller Fahrzeuge bekannt sein.
2. Die Information über das zukünftige Auftreten von Ereignissen ist unpräzise oder unbekannt. Das Auftreten von Ereignissen kann bestenfalls anhand einer Wahrscheinlichkeitsverteilung abgeschätzt werden.
3. Bei Eintreffen der Information von neuen Ereignissen müssen die Datenstrukturen des Lösungsmechanismus aktualisiert werden.
4. Entscheidungen müssen meist schnell getroffen werden. Dies macht kürzere Berechnungszeiten für Optimierungen nötig. Die Größe der Fahrzeugflotte kann oft nicht innerhalb eines kurzen Zeitraums angepasst werden. In diesen Fällen ist der Dispatcher dazu gezwungen, mit der gegebenen Kapazität auszukommen.

[vgl. Larsen u. a., 2007, S.23–25]

2.1.1.5 Dynamic Stochastic Vehicle Routing Problem (DSVRP)

Nach FLATBERG U. A. ist ein Dynamic Stochastic VRP durch folgende Bestandteile definiert:

1. Dem initialen VRP mit den vor der Planung bekannten Eigenschaften.

2. Einer Menge von dynamischen Ereignissen, *Events* genannt, die während der Planung sowie während der Ausführung des Plans auftreten.
3. Probabilistisches Wissen über das zukünftige Eintreten von Ereignissen.
4. Einer *Commitment Strategy*.

Zum ersten Punkt bemerken die Autoren, dass das zugrundeliegende VRP eine reichhaltige Erweiterung des klassischen CVRP ("rich extension of the classical CVRP") sein sollte, um ein möglichst weites Spektrum an realen Anwendungen abzudecken. Bezüglich des zweiten Punkts werden als Beispiele für Events neue Aufträge, die Abänderung von Aufträgen bezüglich Liefermenge oder Zeitfenster sowie die Veränderung von Fahrzeiten aufgrund von Verzögerungen oder verbesserter Prognosen genannt. Als typisches Beispiel für den dritten Punkt werden statistische Daten, gewonnen aus früheren Erfahrungen, genannt [vgl. Flatberg u. a., 2007, S.47–49].

Interessanterweise ist der letzte Punkt der Beschreibung eines DSVRP (Commitment Strategy) nicht Bestandteil des Problems selbst, sondern legt einen Teil der Vorgehensweise zum Lösen des Problems fest. Nach FLATBERG U. A. bestimmt die Commitment Strategy u. a. wann Requests unwiderruflich Fahrzeugen zugeordnet werden und welche Teile eines Plans während der Optimierung abgeändert werden dürfen. [vgl. Flatberg u. a., 2007, S.49]

Bei den veränderlichen Aspekten von Problemen unterscheiden FLATBERG U. A. zwischen dynamischen, stochastischen und zeitabhängigen Aspekten. Dynamische Aspekte verändern sich während der Ausführung des Plans und bedingen eine Änderung des Plans. Bei stochastische Aspekten unterliegt die Wahrscheinlichkeit des Auftretens eines Ereignisses einer bekannten oder unbekannten Wahrscheinlichkeitsverteilung. Demnach sind dynamische Aspekte grundsätzlich auch von stochastischer Art. Zeitabhängige Aspekte beschreiben Daten, deren Wert innerhalb eines Zeitintervalls gültig sind. Als Beispiel nennen die Autoren Fahrtzeiten während der Rush Hour [vgl. Flatberg u. a., 2005, S.3].

2.1.1.6 Abschließende Bemerkung zu den Varianten

Die oben genannten Beispiel von Varianten des Vehicle Routing Problem zeigen, dass sich die Interpretation der Varianten hinsichtlich enthaltener Randbedingungen zwischen verschiedenen Autoren unterscheiden kann. Zudem werden gleiche Varianten von Autoren mit leicht verschiedenen Namen bezeichnet. Ein Bespiel ist das Single Vehicle Pickup and Delivery Problem (SVPDP) [vgl.. Cordeau u. a., Due 2008, S.4], das in der Literatur auch unter dem Namen 1–Pickup and Delivery Problem (1-PDP) [vgl. Savelsbergh und Sol, 1995, S.14] auftaucht. Die Abkürzung einer VRP–Variante kann nicht immer eindeutig ohne den entsprechenden Kontext, in dem die Abkürzung verwendet wird, einer bestimmten VRP–Variante zugeordnet werden. So steht die Abkürzung SDVRP bei ROPKE UND PISINGER für

Site Dependent VRP (bestimmte Requests dürfen aufgrund bestimmter Anforderungen nur von bestimmten Fahrzeugen bedient werden) [vgl. Pisinger und Ropke, 2007, S.5], während bei DROR U. A. die Abkürzung SDVRP für das Split Delivery VRP verwendet wird (Ein Request kann auf mehrere Fahrzeuge verteilt werden, wenn dadurch die Kosten minimiert werden) [vgl. Dror u. a., 1994, S.2].

PISINGER UND ROPKE führen in Pisinger und Ropke [2007] das Rich Pickup and Delivery Problem with Time Windows (RPDPTW) als Basis für die Entwicklung allgemeiner Heuristiken zur Lösung möglichst vieler Routing Probleme in realen Anwendungen ein. Die Autoren stellen die Transformation von den folgenden fünf VRP–Varianten in ein RPDPTW dar: CVRP, VRPTW, SDVRP sowie OVRP (Open VRP – Routen müssen nicht zwingend am Depot enden) und MDVRP (Multiple Depot VRP – es existieren mehrere Depots, die Requests werden zur Bearbeitung jeweils einem Depot zugeordnet) [vgl. Pisinger und Ropke, 2007, S.2–6].

2.1.2 Komplexität von Vehicle Routing Problemen

Das VRP ist ein kombinatorisches Optimierungsproblem und zeichnet sich wie viele Probleme aus diesem Bereich durch seine Hartnäckigkeit aus. Das VRP kann als Generalisierung des Traveling Salesman Problems angesehen werde. Beim Traveling Salesman Problem ist die Aufgabenstellung, von einem Ausgangsort aus eine Anzahl von Orten zu besuchen und danach zum Ausgangsort zurück zukehren, wobei die gesamte zurück gelegte Entfernung minimal sein soll. Graphentheoretisch bedeutet dies, zu einem vorgegebenen Graphen, bei dem alle Ecken je paarweise eine gemeinsame Kante mit nichtnegativen Gewicht besitzen, eine Kantenfolge zu finden, bei der jede Ecke des Graphen den Grad 2 hat und die Summe der Kantengewichte minimal wird. Das TSP kann somit als Spezialfall des VRP angesehen werden, bei dem ein Fahrzeug alle Kunden besucht, wobei keine weiteren Randbedigungen wie Kapazitätsbeschränkung des Fahrzeug oder Zeitfenster gegeben sind.

Das TSP ist von der Komplexitätsklasse NP–schwer [vgl. Klauck und Maas, 1999, S. 120]. Folglich ist auch das VRP ein Problem der Komlexitätsklasse NP–schwer. Savelsbergh zeigte 1985, dass alleine die Erzeugung einer beliebigen, d. h. nicht notwendigerweise optimalen Lösung für das TSP mit Zeitfenstern von der Komplexitätsklasse NP–vollständig ist [vgl. Savelsbergh, 1985, S.299]. Die Optimierung eines VRP gehört somit zu den schwierigsten kombinatorischen Optimierungsproblemen.

2.2 VRP Benchmarks

Im VRP Bereich haben sich zur Einschätzung der Leistungsfähigkeit von Algorithmen und zum Vergleich der Leistungsfähigkeit von Algorithmen verschiedene Benchmarks etabliert. Diese Benchmarks enthalten VRPTW bzw. PDPTW Instanzen, welche meist anhand verschiedener Eigenschaften in Klassen eingeteilt sind. Das Dateiformat der Instanzen folgt meist dem Format des Benchmarks von Marius M. Solomon oder dem Format der VRPLIB von Gerhard Reinelt bzw. ist von einem der beiden Formate abgeleitet. Nachfolgend werden einige der bekanntesten Benchmarks vorgestellt.

2.2.1 Solomon Benchmark

Der Benchmark von Marius M. Solomon beinhaltet sechs Klassen von VRPTW Instanzen, R1, C1 und RC1 sowie R2, C2, und RC2. Der Name einer Instanz setzt sich aus dem Namen der Klasse sowie einer zweistelligen Identifikationsnummer zusammen. Bei mit “R” bezeichneten Klassen ist die geographische Verteilung der Kunden zufallsgeneriert, bei mit “C” bezeichneten Klassen sind die Kunden in Clustern angeordnet. Die Klassen “RC” beinhalten eine Mischung aus Kunden mit zufallsgenerierter geographischer Verteilung und in Clustern angeordneten Kunden. Bei den einzelnen Instanzen ist für das Depot ein Zeitfenster angegeben, wodurch sich der Planungshorizont ergibt. Klassen mit einer “1” im Klassennamen beinhalten Instanzen mit kurzem Planungshorizont, so dass die Routen der einzelnen Fahrzeuge nur relativ wenige, ca. fünf bis zehn Kunden, beinhalten können. Klassen mit einer “2” im Klassennamen beinhalten Instanzen mit langem Planungshorizont, so dass die Routen der einzelnen Fahrzeuge meist mehr als 30 Kunden beinhalten können.

Die Kunden sind jeweils mit laufender Nummer zur Identifikation, ihren euklidischen Koordinaten, dem Bedarf, dem Anfang und Ende des Zeitfenster und dem zur Bearbeitung des Requests nötigen Standzeit angegeben. Das Depot ist zusammen mit den Kunden aufgeführt, die Nummer zur Identifikation sowie Bedarf und Standzeit sind jeweils mit “0” angegeben. Für die Fahrzeuge ist die Anzahl und homogene Kapazität angegeben.

Eine das Dateiformat der Instanzen beschreibende Spezifikation konnte nicht gefunden werden. Das Format ist allerdings sehr einfach und selbstbeschreibend, wie das Beispiel in Tabelle 2.2.1 zeigt.

Die Instanzen des Benchmarks können aus dem Internet von der Adresse http://w.cba.neu.edu/~msolomon/problems.htm heruntergeladen werden. Auf der gleichen Seite befinden sich auch Links zu bekannten optimalen Lösungen und zu besten bekannten Lösungen, die mit Heuristiken erzielt wurden, wobei die letzte Aktualisierung der Seite am 24. März 2005 stattfand.

RC101

Vehicle

Number	Capacity
25	200

Customer

Cust. No	Xcoord.	Ycoord.	Demand	Ready Time	Due Date	Service Time
0	40	50	0	0	240	0
1	25	85	20	145	175	10
⋮	⋮	⋮	⋮	⋮	⋮	⋮
100	31	67	3	180	210	10

Tabelle 2.1: Auszug aus Instanz rc101 des Solomon Benchmarks.

2.2.2 VRPLIB von Reinelt

Das Format der VRPLIB von Gerhard Reinelt ist um einiges komplexer als das Format des Solomon Benchmarks. Die Position von Kunden kann in euklidischen Koordinaten oder Geokoordinaten angegeben werden. Kantengewichte (Distanzen zwischen Kunden) können auf verschieden Art definiert sein, u. a. als euklidische Distanzen, Manhattan–Distanz oder explizit als Kantengewichts–Matrix. Für Fahrzeuge wird die (uniforme) Kapazität und für Kunden der Bedarf angegeben. Für das Format existiert eine detaillierte Spezifikation, abrufbar von der Internet–Präsenz der Universität Heidelberg unter der Internet–Adresse http://www.informatik.uni-heidelberg.de/groups/comopt/software/TSPLIB95/DOC.PS. Die Instanzen des Benchmarks sind unter der Adresse http://www.iwr.uni-heidelberg.de/groups/comopt/software/TSPLIB95/vrp/ abrufbar.

Da VRP–Instanzen der VRPLIB von Reinelt keine Zeitfenster beinhalten, wird diese VRPLIB kaum als Benchmark im VRP Bereich benutzt. Die Spezifikation bietet jedoch eine gute Grundlage zur Erweiterung auf Randbedingungen wie Zeitfenster. Einige der im folgenden beschriebenen Benchmarks sind durch eine Erweiterung der Spezifikation der VRPLIB von Reinelt definiert.

2.2.3 Benchmark von Li und Lim

Der Benchmark von LI UND LIM beinhaltet PDPTW Instanzen, die nach dem Schema von Solomon (R1, R2, C1, C2, sowie RC1 und RC2) klassifiziert sind [vgl. Li und Lim, 2003, S.5]. Die Instanzen sind zusätzlich nach Anzahl der Kunden in Klassen zu 100, 200, 400, 600,

```
Name: eil22
Comment: (Eilon et al.)
Type: CVRP
Dimension: 22
Edge_Weight_Type: EUD_2D
Capacity: 6000
Node_Coord_Section
 1        145        215
 2        151        264
 ⋮         ⋮          ⋮
22        139        182
Demand_Section
 1          0
 2       1100
 ⋮          ⋮
22        700
Depot_Section
 1
-1
EOF
```

Tabelle 2.2: Auszug aus Instanz eil22 der VRPLIB von Reinelt.

800 und 1000 Kunden klassifiziert, wobei die Klassen nach Anzahl der Kunden jeweils 58 – 60 Instanzen beinhalten. Innerhalb des Benchmarks sind die Instanzen nach Klassen der Anzahl von Kunden sortiert. Die Spezifikation für die Instanzen ist einfach. Für jede Instanz ist die maximale Anzahl an Fahrzeugen sowie die uniforme Kapazität und Geschwindigkeit der Fahrzeuge definiert. Nach Angaben in der Spezifikation wird allerdings der Wert für die Geschwindigkeit nicht benutzt. Die Kunden werden jeweils durch einen Datensatz in einer Zeile definiert. Die Daten bestehen aus einer laufenden Nummer zur Identifikation des Kunden, den euklidischen Koordinaten, die zur Abholung bzw. Belieferung notwendige Kapazität, Anfang und Ende des Zeitfensters, Standzeit, sowie die Identifikationsnummern der Kunden, die den Pickup–Punkt und den Delivery–Punkt darstellen. Bei den letzten beiden Einträgen ist jeweils für einen Eintrag die Menge “0” angegeben, wodurch definiert ist, ob es sich bei dem Kunden um einen Pickup–Punkt oder um einen Delivery–Punkt handelt. Handelt es sich bei einem Kunden um einen Pickup–Punkt, ist die Kapazität mit positivem Wert angegeben. Bei einem Delivery–Punkt ist dieser Wert negativ. Der Definition der Fahrzeugeigenschaften folgen stets die Angaben für das Depot, wobei die Werte für laufende Nummer zur Identifikation, Kapazität, Standzeit sowie Pickup– und Delivery–Punkt beim Depot stets “0” betragen. Die Dateien der Instanzen enthalten keine Spaltenüberschrften. Tabelle 2.2.3 zeigt einen

Auszug aus der Instanz lc102 mit einem Request, bei dem u. a. eine Ladung der Kapazität “10” von Kunde 1 zu Kunde 75 zu transportieren ist.

25	200	1						
0	40	50	0	0	1236	0	0	0
1	45	60	10	0	1127	90	0	75
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
75	45	65	-10	0	1130	90	1	0
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
106	26	32	-10	0	1123	90	50	0

Tabelle 2.3: Auszug aus Instanz lc102 des Benchmarks von Li und Lim

Die Dateinamen der Instanzen folgen dem von Solomon verwendeten Schema, zusätzlich ist der erste Buchstabe des Dateinamens stets ein “l”, um eine Verwechslung mit Instanzen aus dem Solomon Benchmak auszuschliessen. Allerdings werden je nach Klasse für die Anzahl an Kunden entweder Klein– oder Grossbuchstaben verwendet, die Durchnummerierung der Instanzen ist ebenfalls nur innerhalb einer Klasse für die Anzahl an Kunden konsistent.

Die Instanzen und eine Beschreibung des Formats stehen unter der Internet–Adresse http://www.sintef.no/static/am/opti/projects/top/vrp/benchmarks.html zum Download bereit (URL zuletzt geprüft am xx.yy.zzzz).

2.2.4 Benchmark von Ropke und Pisinger

STEFAN ROPKE verwendet in Ropke und Pisinger [2006a] sowohl den Benchmark von Li und Lim als auch einen von Ropke und Pisinger erstellten PDPTW–Benchmark. Die Spezifikation des von Ropke und Pisinger erstellten Benchmarks ist dem Benchmark von Li und Lim sehr ähnlich, enthält jedoch einige Zusätze.

Am Anfang der Instanz–Datei sind die Anzahl der Requests (bestehend aus einem Pickup– und Delivery–Punkt, die maximale Anzahl an Fahrzeugen, sowie die drei Parameter α, β und γ der Bewertungsfunktion. Die Parameter α und β beschreiben die Gewichtung der Kosten hinsichtlich der zeitlichen Dauer und Streckenlänge einer Lösung. Der Parameter γ ist ein “Strafwert”, der für in der Lösung nicht berücksichtigte Requests auf die Kosten der Lösung aufgeschlagen wird. Dieser Wert wird meist entsprechend hoch definiert (in den Instanzen beträgt der Wert 100000 gegenüber 1 für α und β), so das alle Requests berücksichtigt werden.

Für die Fahrzeuge werden jeweils eine laufende Nummer zur Identifikation und die Depots, von denen die Routen der Fahrzeuge starten bzw. enden, mit Koordinaten der betreffenden

50	15	1	1	1	100000				
0	538	102	15	2406	2945	180	-1	1	-1
1	755	194	-15	3537	4283	153	0	-1	
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
99	30	582	-19	3806	4650	167	98	-1	
0	852	331	852	331	50	0	5000		
1	819	858	819	858	50	0	5000		
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮		
14	552	210	552	210	50	0	5000		

Tabelle 2.4: Auszug aus Instanz prob50A des Benchmarks von Ropke und Pisinger

Depots angegeben. Damit ist grundsätzlich die Erfassung von sowohl von PDP als auch MDPDP (Multiple Depot PDP) möglich. Die Instanzen des Benchmarks sind alle vom Typ MDPDP. Des weiteren wird für jedes Fahrzeug gesondert die Kapazität und das Zeitfenster, während dem das Fahrzeug zur Verfügung steht, angegeben. Die Angaben für die Fahrzeuge werden in einem Abschnitt am Ende der Instanz–Datei angegeben. Für einen Pickup–Punkt kann definiert werden, welche Fahrzeuge zur Abholung von Gütern geeignet sind, die betreffenden Nummern der Fahrzeuge werden dazu an den Datensatz eines Kunden angehängt. Sind alle Fahrzeuge zur Abholung von Gütern bei einem Kunden geeignet, wird dies durch Anfügen des Wertes “-1” an den betreffenden Kunden–Datensatz angezeigt. In der Spezifikation wird nicht die Angabe von Kunden wie beim Benchmark von Li und Lim definiert, stattdessen werden die Angaben für Pickup– bzw. Delivery–Punkte (“node-id”) definiert. Ansonsten entspricht das Format eines Datensatzes für Pickup– bzw. Delivery–Punkte dem Format des Benchmarks von Li und Lim. Die Dateien der Instanzen enthalten keine Spaltenüberschrften. Ein Auszug aus einer Instanz des Benchmarks wird in Tabelle 2.2.4 gezeigt.

Der Benchmark enthält 48 Instanzen, wobei je 12 Instanzen jeweils 50, 100, 250 bzw. 500 Kunden beinhalten. Der Name einer Instanz setzt sich aus dem Anfang “prob”, gefolgt von der Anzahl der Requests innerhalb der Instanz sowie einem Grossbuchstaben von “A” bis “L” zusammen.

Der Benchmark wird von der Internet–Präsenz der Universität Kopenhagen, Computer Science Department, auf der persönlichen Seite von Stefan Ropke unter der Adresse http://www.di.ku.dk/ sropke/DataSets/PDPTWInstances.zip zum Download bereitgestellt (URL zuletzt geprüft am 12.08.2008). s

2.3 Verfahren zur Lösung von Vehicle Routing Problemen

Aufgrund der Komplexität von Vehicle Routing Problemen und der damit verbundenen kombinatorischen Explosion bei wachsender Eingabegröße ist eine Evaluation aller im *Suchraum* bzw. *Lösungsraum* enthaltenen Lösungen nur bei trivialen Problemen praktikabel. Als Suchraum wird die Menge aller Lösungen eines Problems aufgefasst, die von einem zur Lösung des Problmes eingesetzten Algorithmus durchsucht wird. Als Beispiel ist bei einem TSP der Suchraum die Menge aller möglichen Permutationen über die Menge der Ecken des dem TSP zugrundeliegenden Graphen. Der Lösungsraum ist die Menge aller gültigen Lösungen. Bei einem TSP ohne Zeitfenster oder sonstige Randbedingungen ist obiger Suchraum identisch mit dem Lösungsraum. Sind Randbedingungen wie z. B. die Reihenfolge des Auftretens bestimmter Ecken in einer Lösung definiert, ist dies nicht der Fall. Als *Nachbarschaft* einer Lösung L werden diejenigen Lösungen bezeichnet, die durch geringfügige Modifikation der Lösung L generiert werden können. Bei Optimierungsproblemen sind meist mehrere Nachbarschaften möglich, wobei verschiedene Nachbarschaften eine gemeinsame Teilmenge besitzen können. Bei einem TSP besteht eine Nachbarschaft einer Lösung L aus den Lösungen, die von der Lösung L ausgehend durch Verschieben einer Ecke bzw. einiger wenigen Ecken generiert werden können. Eine andere Nachbarschaft besteht in diesem Fall aus denjenigen Lösungen, die aus L durch Entfernung einiger Kanten mit anschliessender Verbindung der Teiltouren durch Einfügen neuer Kanten generiert werden können. Die hier beschriebenen Verfahren stellen Methoden zur Suche in einem Suchraum bzw. Lösungsraum dar und zielen darauf ab, durch die Evaluation einer möglichst geringen Anzahl von Lösungen eine möglichst optimale gültige Lösung zu finden. Dadurch wird versucht, die Problematik der kombinatorischen Explosion zu umgehen.

2.3.1 Einteilung der Verfahren

Die im VRP Bereich verwendeten Verfahren lassen sich zunächst in exakte Verfahren und Heuristiken aufteilen. Die exakten Verfahren zielen darauf ab, iterativ im Suchraum Bereiche auszumachen, in denen weder eine gültige noch eine optimale Lösung enthalten sein kann, um so den Suchraum zu verkleinern. Die dazu verwendeten Techniken sind sehr aufwendig und kompliziert. Trotzdem lassen sich mit exakten Methoden bisher nur vergleichsweise kleine Probleminstanzen lösen. Da für die Problemstellung die Anwendung exakter Methoden nicht in Frage kommt [1], werden diese nur kurz angerissen. Das Wort Heuristik stammt aus dem griechischen *heuriskein* ($\epsilon\upsilon\rho\iota\sigma\kappa\iota\nu$) und bedeutet “ich finde” bzw. “ich entdecke”. NEWELL U. A. definierten 1957 Heuristik als “a process, that may solve a problem, but offers no guarantess of doing so” [Newell u. a., 1957; zit. n. Wah u. a., 1995, S.1]. Die heuristischen

[1] zur Begründung siehe Kapitel 2.3.2 (Exakte Verfahren)

Verfahren zur Lösung von Vehicle Routing Problemen lassen sich in klassische Heuristiken und Metaheuristiken einteilen. LAPORTE U. A. ordnen die klassischen Heuristiken in die Periode von 1960 bis 1990 ein, während die Metaheuristiken im Bereich des VRP ab 1990 Bedeutung erlangen [vgl. Laporte u. a., 2000, S.1].

Bei der Durchsicht bestehender Literatur zu klassischen Heuristiken aus dem VRP Bereich fallen zunächst zwei Klassen auf, in die sich die Heuristiken einteilen lassen: Heuristiken zur Konstruktion initialer Lösungen und Methoden, die eine bestehende Lösung modifizieren und zur Suche im Suchraum bzw. Lösungsraum dienen.

In ihrer Übersicht über klassische Heuristiken für das VRP unterscheiden LAPORTE UND SEMET in [Laporte und Semet, 2001, S.1] drei Arten von klassischen Heuristiken: *Konstruktionsheuristiken*, *zwei–Phasen Heuristiken* und *Verbesserungsmethoden*. *Konstruktionsheuristiken* generieren unter Berücksichtigung der Kosten eine gültige Lösung, beinhalten aber *per se* keine anschließende Phase zur Verbesserung der generierten Tour. Bei den *Zwei–Phasen Heuristiken* wird das Problem in zwei Bestandteile zerlegt: dem Zusammenfassen geographisch naheliegender Ecken zu sogenannten Clustern, und dem Lösen eines TSP für die jeweiligen Cluster. Die Autoren erwähnen, dass zwischen den beiden Phasen ein Feedback stattfinden kann. Die zwei–Phasen Heuristiken werden in zwei Klassen unterteilt: Cluster–First, Route–Second Methoden und Route–First, Cluster–Second Methoden. Bei ersteren Methoden werden die Kunden in Clustern zusammengefasst, die eine gültige Lösung erlauben und danach für jeden Cluster ein TSP gelöst. Bei letzteren Methoden wird zuerst für alle Ecken des Graphen ein TSP gelöst. Die entstandene Rundreise wird in gültige Teilrouten aufgeteilt. *Verbesserungmethoden* verändern eine oder mehrere Routen einer bestehenden gültigen Lösung, indem Ecken oder Kanten innerhalb einer Routen vertauscht bzw. zwischen mehreren Routen ausgetauscht werden.

Nach Ansicht des Autors können jedoch auch die zwei-Phasen Heuristiken zu den Konstruktionsheuristiken gezählt werden, da beide Verfahren die Ecken eines Graphen zu Routen zusammenfassen. Dass Konstruktionsheuristiken *per se* keine Optimierung der generierten Routen vornehmen, spricht dem nicht entgegen, da die in den zwei-Phasen Heuristiken verwendeten Optimierungsverfahren auch innerhalb von Konstruktionsheuristiken verwendet werden können.

Bei Metaheuristiken wird das Verfahren zur Suche im Such– bzw. Lösungsraum durch einen übergeordneten Prozess gesteuert (*meta*, griechisch, “auf übergeordneter Ebene”). Dabei werden Informationen über den Such– bzw. Lösungsraum, die während des Suchprozesses gewonnen werden, dazu benutzt, die Suche möglichst effizient zu gestalten. Daher werden metaheuristische Verfahren dem Bereich der künstlichen Intelligenz zugeordnet.

Da viele Metaheuristiken eine bestehende – jedoch nicht notwendigerweise qualitativ gute Lösung – voraussetzen, um diese dann iterativ zu verbessern, werden oft klassische Konstruktionsheuristiken als Generatoren für Startlösungen eingesetzt. Bei der Suche im Such-

raum kommen bei Metaheuristiken meist klassische Modifikationsoperatoren, bisweilen in erweiterter Form, zum Einsatz. Daher haben die klassischen Heuristiken aus dem VRP Bereich nach wie vor Bedeutung.

2.3.2 Exakte Verfahren

Aus dem Bereich der exakten Algorithmen werden nach ROPKE UND CORDEAU zur Lösung von PDPTW hauptsächlich die Algorithmen Branch–and–Price und Branch–and–Cut verwendet [vgl. Ropke und Cordeau, 2007, S.1]. Beide Verfahren sind Weiterentwicklungen des Branch–and–Bound Verfahrens. Die folgende Beschreibung der Branch-and-Bound Verfahren basiert auf den Ausführungen von V. KUMAR in der *Encyclopedia of Artificial Intelligence*.

Branch–and–Bound Verfahren wurden in den 1960er Jahren zur Behandlung von ganzzahligen (linearen) Optimierungsproblemen und nicht–linearen Zuweisungproblemen entwickelt. Der Grundansatz des Verfahrens besteht darin, den Such– bzw. Lösungsraum iterativ in Teilräume zu zerlegen. Beim Depth–First Branch–and–Bound wird derjenige Teilraum weiter zerlegt, der zuletzt durch die Zerlegung erzeugt wurde. Beim Best–First Branch–and–Bound wird für jeden Teilraum X_i eine untere Schranke $lb(X_i)$ bestimmt, wobei $f(x) \geq lb(X_i)$ für alle $x \in X_i$. Der Teilraum mit der kleinsten unteren Schranke wird jeweils weiter in immer kleinere Teilräume zerlegt, bis der Teilraum mit der kleinsten unteren Schranke aus einer Lösung besteht. In diesem Fall terminiert das First–Best Branch–and–Bound.

Die Prozedur zur Aufteilung eines Teilraums hängt von der Art des Problems ab. Können in einem Lösungsraum L zwei Teilräume X_1 und X_2 identifiziert werden, für die gilt: $\forall x_1 \in X_1, x_2 \in X_2 : f(x_1) \geq f(x_2)$, dann kann der Teilraum X_1 abgeschnitten werden, d. h. die Lösungen des Teilraums X_1 werden nicht mehr weiter betrachtet. Dies wrd als *Pruning* bezeichnet Kumar [1992].

Die Leistungsfähigkeit von Branch-and-Bound Methoden zur Lösung von VRP ist begrenzt und hängt zudem stark von den gegebenen Randbedingungen des zu lösenden Problems ab. Cordeau löste 2006 mit einem von ihm entwickelten Branch–and-Cut Algorithmus branch-and-cut Algorithmus DARP–Instanzen mit vier Fahrzeugen und bis zu 32 Requests [vgl. Cordeau, 2006]. Ropke, Cordeau und Laporte gelang es 2007, DARP–Instanzen mit acht Fahrzeugen und 96 Requests zu lösen [vgl. Ropke u. a., 2007, S.1].

In Ropke und Cordeau [2007] vergleichen ROPKE UND CORDEAU einen von ihnen entwickelten Branch–and–Cut–and–Price Algorithmus mit einer von Ropke und Pisinger entwickelten Metaheuristik, dem Advanced Large Neighbourhood Search (ALNS). Zum Vergleich wurden 40 Testinstanzen herangezogen. Die Testinstanzen sind bezüglich der für sie definierten Randbedingungen in Form von Fahrzeugkapazität und Zeitfensterbreite in vier Klassen

von je zehn Testinstanzen eingeteilt. Die vier Klassen werden mit AA, BB, CC und DD bezeichnet. Die (abstrakte) Fahrzeugkapazität beträgt für diese Klassen 15, 20, 15 und 20. Die Zeitfensterbreite beträgt 60, 60, 120 und 120 bei einem Planungshorizont von 600. Die Nachfrage eines Requests wird bei den Instanzen vorab zufallsgesteuert aus dem Intervall $[5, Q]$ bestimmt, wobei Q die Fahrzeugkapazität der Instanzklasse ist. [vgl. Ropke und Cordeau, 2007, S.28]. Der Name einer Testinstanz setzt sich aus der Bezeichnung der Klasse und der Anzahl der Requests zusammen. So beträgt z. B. bei der Instanz CC30 die Fahrzeugkapazität 15, die Zeitfensterbreite beträgt 120 und es sind 30 Requests vorhanden. Oberstes Ziel bei der Optimierung bei allen Instanzen ist die Minimierung der Anzahl der benötigten Fahrzeuge [vgl. Ropke und Cordeau, 2007, S.27]. In den tabellarischen Auflistungen der Versuchsergebnisse sind diese jedoch nicht angegeben, dafür allerdings die jeweiligen Laufzeiten. Für den Branch–and–Cut–and–Price Algorithmus wird die Laufzeit nur angegeben, wenn diese unter Einhaltung bestimmter algorithmus–spezifischer Bedingungen nicht mehr als zwei Stunden beträgt. Andernfalls wird die Instanz in diesem Fall als nicht gelöst bewertet [vgl. Ropke und Cordeau, 2007, S.29, 40]. Die algorithmus–spezifischen Bedingungen konnten vom Autor mangels fundierter Kenntnisse im Bereich Branch–and–Bound nicht nachvollzogen werden. Die Testergebnisse verdeutlichen die Überlegenheit des ALNS über den Branch–and–Cut–and–Price Algorithmus. Während von ALNS alle Testinstanzen mit einer Laufzeit unter sieben Minuten gelöst werden, löst der Branch–and–Cut–and–Price Algorithmus nur 12 der 40 Testinstanzen innerhalb des Zeitlimits [vgl. Ropke und Cordeau, 2007, S.37, 40]. Aus der Klasse AA löst der Branch–and–Cut–and–Price Algorithmus die acht Instanzen mit 30 bis 65 Requests und schlägt bei den Instanzen mit 30, 35, 40 und 55 Requests ALNS hinsichtlich der Laufzeit. Bei den übrigen vier gelösten Instanzen variiert die Laufzeit des Branch–and–Cut–and–Price Algorithmus stark. Bei der Instanz AA45 beträgt sie ca. 25 Minuten während ALNS diese Instanz innerhalb von ca. 2 Minuten löst. Aus der Klasse BB löst der Branch–and–Bound Algorithmus vier Instanzen innerhalb des Zeitlimits, mit bis zu maximal 50 Requests. Aus der Klasse CC und DD kann der Branch–and–Bound Algorithmus keine Instanz innerhalb der gegebenen zwei Stunden lösen [vgl. Ropke und Cordeau, 2007, S.27].

Die Ergebnisse zeigen, dass Branch–and–Bound bei Instanzen mit einer kleinen Anzahl von Requests und starken Einschränkungen des Lösungsraums durch Randbedingungen sehr leistungsfähig ist. Erhöht sich jedoch die Anzahl an Requests oder fehlen starke Einschränkungen des Lösungsraums durch Randbedingungen, fällt die Performanz ab. Für die Aufgabenstellung dieser Arbeit relevante Szenarios werden eher durch Instanzen der Klasse BB, CC und DD dargestellt. Aus diesem Grund erscheint die Verwendung von Branch–and–Bound Verfahren zur Lösung der Aufgabenstellung als nicht empfehlenswert.

2.3.3 Klassische Konstruktionsheuristiken

LAPORTE UND SEMET beschreiben drei Konstruktionsheuristiken, wobei für eine Heuristik, den *Clarke und Wright Savings Algorithmus* (benannt nach seinen Urhebern, G. Clarke und J.W. Wright) vier Varianten angegeben werden. Desweiteren werden fünf Cluster–First Route–Second Methoden und die Grundzüge der Route–First Cluster–Second Methode beschrieben. Alle Methoden bis auf die Route–First Cluster–Second Methode werden anhand von 14 Testinstanzen aus der VRPLIB von Vigo hinsichtlich Lösungsqualität und Laufzeit auf ihre Performanz untersucht [vgl. z. B. Laporte und Semet, 2001, S.6, 12]. LAPORTE UND SEMET geben an, das ihnen keine Route–First Cluster–Second Methode bekannt ist, deren Performanz an die der anderen Verfahren heranreicht [vgl. Laporte und Semet, 2001, S.13]. Die verwendeten Testinstanzen sind CVRP, von denen einige als Randbedingung eine Beschränkung der maximalen Länge aufweisen. Weitere Randbedingungen sind nicht vorhanden. Werden die Ergebnisse der Tests hinsichtlich der Eignung der Verfahren als Generatoren für Startlösungen für Metaheuristiken interpretiert, ergeben sich keine nennenswerten Unterschiede. Während die komplexeren Algorithmen einige Prozent–Punkte hinsichtlich der Lösungsqualität weiter vorne liegen, liegen die Ergebnisse aller Algorithmen bei allen Testinstanzen durchweg weniger als 10% über dem bekannten Minimum der jeweiligen Testinstanz. Für Startlösungen von Metaheuristiken ist diese Qualität ausreichend. Ein wichtiges Kriterium zur Eignung einer klassischer Heuristik als Generator für Startlösungen von Metaheuristiken ist die Möglichkeit einer relativ einfachen Erweiterung, durch die der Generator in der Lage ist, Randbedingungen von realen VRP zu berücksichtigen. Hier sind die einfacheren klassischen Heuristiken klar im Vorteil.

In ihren Schlussbemerkungen führen LAPORTE UND SEMET an, dass sich mehrere der untersuchten Konstruktionsheuristiken leicht auf andere VRP–Varianten anpassen lassen. Der Clarke und Wright Savings Algorithmus wird in diesem Zusammenhang als wahrscheinlich populärster Algorithmus genannt [vgl. Laporte und Semet, 2001, S.17].

Im folgenden werden je eine von LAPORTE UND SEMET beschriebene Konstruktionsheuristik und zwei-Phasen Heuristik sowie mehrere Verbesserungsmethoden vorgestellt. Die Darstellung der Verbesserungsmethoden fällt bei Laporte und Semet 2001 etwas knapp aus. Die Darstellung der hier vorgestellten Verbesserungsmethoden ist der ausführlichen Darstellung in Kindervater und Savelsbergh [1997] entnommen.

2.3.3.1 Der Clarke und Wright Savings Algorithmus

Die wohl bekannteste Konstruktionsheuristik basiert auf der Idee der Einsparung beim Zusammenfügen zweier Routen in eine neue Route. Werden zwei Routen mit den Ecken

$(0, \ldots, i, 0)$ und $(0, j, \ldots, 0)$ zu einer neuen Route mit den Ecken $(0, \ldots, i, j, \ldots, 0)$ zusammengefasst, ergibt sich eine Einsparung an Kantengewicht vom Betrag $s_{ij} = c_{i0} + c_{0j} - c_{ij}$, wobei die Ecke 0 das Depot und c_{mn} die Kosten der Kante mit Startecke m und Endecke n darstellt. Von diesem Algorithmus existiert eine sequentielle und eine parallele Variante. Die Funktionsweise des Algorithmus wird im folgenden beschrieben:

Parallele Version (Best feasible merge)

1. Berechne die Einsparungen $s_{ij} = c_{i1} + c_{1j} - c_{ij}$ für alle $i, j = 1, \ldots, n$ und $i \neq j$. Sortiere die Einsparungen in aufsteigender Reihenfolge. Erzeuge n Routen $(0, i, 0)$ für $i = 1, \ldots, n$.
2. Entferne das erste Element s_{ij} aus der Einsparungsliste.
3. Wenn für die Einsparung s_{ij} zwei Routen existieren, von denen jeweils eine die Kante $(i, 0)$ und eine die Kante $(0, j)$ besitzen, gehe zu 4. Ansonsten gehe zu 2.
4. Wenn beim Zusammenfügen der beiden Routen durch Entfernung der Kanten $(0, j)$ und $(i, 0)$ und Einführung der Kante (i, j) eine gültige Lösung entsteht, dann füge beide Routen zusammen. Wenn die Einsparungsliste leer ist, dann terminiere. Ansonsten gehe zu 2.

Sequentielle Version (Route extension)

1. Berechne die Einsparungen $s_{ij} = c_{i1} + c1j - cij$ für alle $i, j = 1, \ldots, n$ und $i \neq j$. Sortiere die Einsparungen in aufsteigender Reihenfolge. Erzeuge eine Liste mit n Routen $(0, i, 0)$ für $i = 1, \ldots, n$.
2. Entferne die erste Route aus der Routenliste.
3. Bestimme, ob für die aktuelle Route $(0, i, \ldots, j, 0)$ eine zweite Route $(0, k, \ldots, l, 0)$ besteht, so dass die durch Zusammenfügen der beiden Routen enstehende Route $(0, i, \ldots, j, k, \ldots, l, 0)$ gültig ist. Wenn ja, dann füge die beiden Routen zusammen, entferne die zweite Route aus der Routenliste und gehe zu 3.
4. Wenn die Routenliste leer ist, dann terminiere. ansonsten gehe zu 1.

Bei dem von Laporte und Semet durchgeführten Test der parallelen und sequentiellen Variante erzeugt die parallele Variante bei allen Testinstanzen bessere Lösungen als die sequentielle Variante [vgl. Laporte und Semet, 2001, S.3]. Ein Nachteil beider Varianten ist, dass zu Anfang gute Routen erzeugt werden, im späteren Verlauf jedoch sehr schlechte Routen entstehen.

2.3.3.2 Der Sweep Algorithmus

Der Sweep Algorithmus ist der einfachste Zwei–Phasen Algorithmus in der Übersicht von Laporte und Semet. Die Arbeitsweise des Algorithmus lässt sich am besten anhand eines Radarstrahls veranschaulichen, der vom Depot ausgehend nacheinander alle Ecken des Graphen überstreicht. Die Ecken werden durch Polarkoordinaten (θ_i, ρ_i) repräsentiert, wobei jeweils der Winkel durch θ_i und die Länge durch ρ_i bestimmt wird. Für eine beliebige Ecke i^* gelte $\theta_i^* = 0$. Für alle anderen Ecken wird der Winkel von der Kante $(0, i^*)$ berechnet. Die Ecken werden nach ihren Winkeln in aufsteigender Reihenfolge sortiert in eine Liste eingefügt. Danach werden die folgenden Schritte ausgeführt:

1. (Routeninitialisierung). Wähle ein unbenutztes Fahrzeug k.
2. (Routenkonstruktion). Ordne dem Fahrzeug k iterativ das jeweils erste Element aus der Eckenliste zu, solange durch die Zuordnung keine Randbedingung (z. B. Fahrzeugkapazität, maximale Routenlänge, etc.) verletzt wird. Zugeordnete Ecken werden aus der Eckenliste entfernt. Nach der Zuordnung einer Ecke kann wahlweise eine Optimierung der Route vorgenommen werden.
3. (Routenoptimierung). Optimiere die Route des Fahrzeugs k.
4. Wenn die Eckenliste leer ist, terminiere. Ansonsten gehe zu 1.

2.3.4 Modifikationsoperatoren

Die Standardoperatoren zur Modifikation einer bestehenden Lösung eines VRP sind der k–opt Operator und Weiterentwicklungen dieses Operators. Die Beschreibung der folgenden Operatoren ist der Übersicht von KINDERVATER UND SAVELSBERGH Kindervater und Savelsbergh [1997] entnommen. Informationen, die aus anderen Quellen stammen, sind durch die entsprechenden Quellenbelege gekennzeichnet.

2.3.4.1 Der k–opt Operator

Der k–opt Operator entfernt aus einer TSP Tour k Kanten und fügt die dabei entstehenden Teiltouren durch Einfügen neuer Kanten zu einer neuen TSP Tour zusammen, wobei üblicherweise $2 \leq k \leq 3$. Das Zusammenfügen der Teiltouren kann deterministisch oder zufallsgesteuert erfolgen. Abbildungen 2.1 und 2.2 veranschaulichen k–opt Modifikation für $k = 2$ und $k = 3$.

Eine TSP Tour ist k–optimal, wenn durch Anwendung des k–opt Operators keine Verbesserung der Tour möglich ist. Die Überprüfung, ob eine TSP Tour k–optimal ist, hat die Komplexität $O(n^k)$ [vgl. u. a. Laporte und Semet, 2001, S.13]. Der k–opt Operator ist auch unter den Namen k–opt Echxange [vgl. u. a. Kindervater und Savelsbergh, 1997, S.2] bzw. λ–opt Operator [vgl. u. a. Laporte und Semet, 2001, S.13] bekannt. Der k–opt Operator wird auch oft als Lin–Kernighan–Operator bezeichnet. JOHNSON nennt als Ursprung des 2–opt die Arbeiten von Flood [1956] sowie Croes [1958], für den 3–opt werden die Arbeiten von BOCK (1958) [2] sowie Lin [1965] genannt [vgl. Johnson und McGeoch, 1997, S.16].

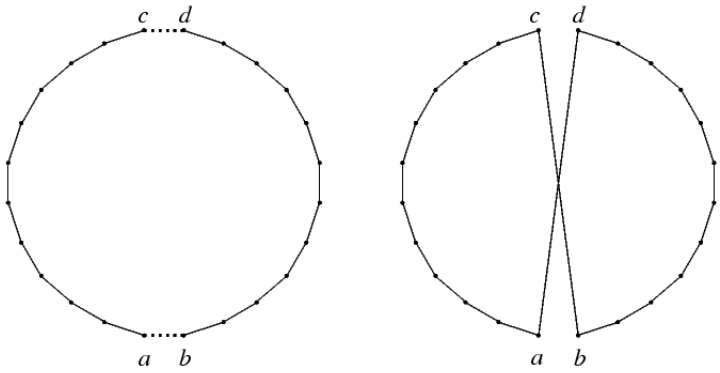

Abbildung 2.1: 2–opt Operator: originale Tour auf der linken und resultierende Tour auf der rechten Seite. Graphik aus Johnson und McGeoch [1997]

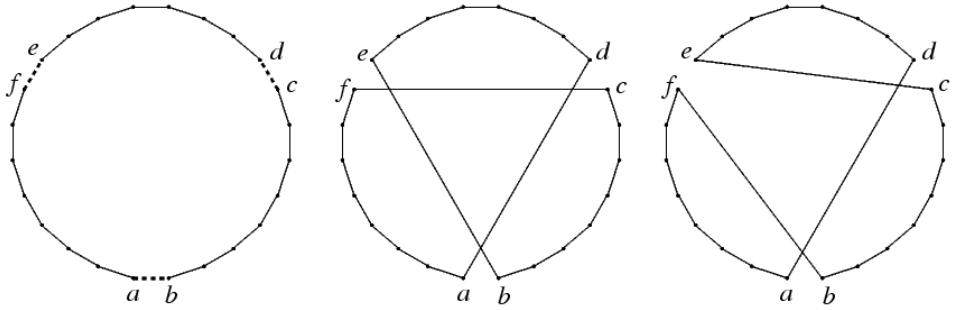

Abbildung 2.2: 3-opt Operator: originale Tour auf der linken und zwei Möglichkeiten von resultierenden Touren auf der rechten Seite. Graphik aus Johnson und McGeoch [1997]

[2] Die Arbeit von Bock stellt ein unveröffentlichtes Manuskript im Zusammenhang mit einer Präsentation während des 14. ORSA National Meetings 1958 dar. Die Zitierwürdigkeit wird deshalb als fragwürdig eingestuft.

2.3.4.2 Der Or–opt Operator

Nach KINDERVATER UND SAVELSBERGH wurde der Or–opt Operator wurde von I.OR in seiner Dissertationsarbeit Or [1976] eingeführt [vgl. Kindervater und Savelsbergh, 1997, S.3].

Beim Or–opt Operator wird eine Folge von l Ecken innerhalb einer TSP–Tour "verschoben": sei $(0, \ldots, i, k, \ldots, p, q, \ldots, u, v, \ldots, 0)$ eine TSP–Tour und $(k, \ldots, p)$ die Folge von Ecken, die zunächst aus der TSP–Tour entfernt wird, um sie danach zwischen den Ecken u und v wieder einzufügen. Dazu werden die Kanten (i, k) und (p, q) aus der TSP–Tour entfernt und die Ecken i und q durch Einfügen der Kante (i, q) miteinander verbunden. Die dadurch entfernte Folge von Ecken $(k, \ldots, q)$ wird in die TSP–Tour wieder eingefügt, indem die Kante (u, v) entfernt wird und die Kanten (u, k) sowie (p, v) eingeführt werden. Abbildung 2.3 veranschaulicht die or–opt Operation für $l = 1$.

KINDERVATER UND SAVELSBERGH heben hervor, das die Nachbarschaft des Or–opt Operators eine Teilmenge der Nachbarschaft des k-opt operators darstellt [vgl. Kindervater und Savelsbergh, 1997, S.3,4].

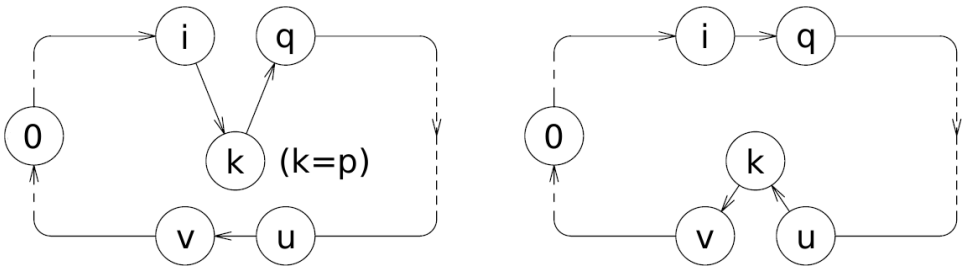

Abbildung 2.3: Or–opt Operator. Graphik nach Kindervater und Savelsbergh [1997]

2.3.4.3 Node Relocation und Path Relocation

Zur Modifikation von Lösungen eines VRP kann der Or–opt Operator in der Art angewendet werden, dass eine Ecke bzw. eine Folge von Ecken aus der Route eines Fahrzeugs entfernt wird und in die Route eines anderen Fahrzeugs eingefügt wird. Die Operationen zum Enfernen und Wiedereinfügen der Ecke bzw. der Folge von Ecken sind analog zum or-opt Anwendung auf eine TSP–Tour, mit dem Unterschied, dass die Operation zum Entfernen bzw. Einfügen auf je eine Tour eines Fahrzeugs angewendet wird. KINDERVATER UND SAVELSBERGH beschreiben die Operationen als *Relocation* und *Relocation of a Path*. In dieser Arbeit wird der Operator bei $l = 1$ als *Node Relocation* bezeichnet, da dies den Sachverhalt eindeutig beschreibt. Die Abbildungen 2.4 und 2.5 veranschaulichen die Operationen des Node Relocation und Path Relocation.

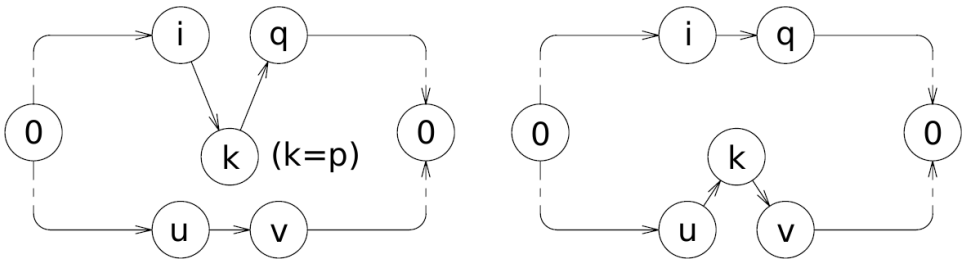

Abbildung 2.4: Node Relocation. Graphik nach Kindervater und Savelsbergh [1997]

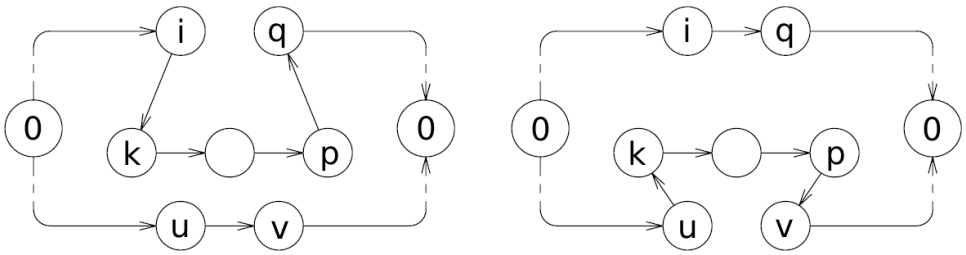

Abbildung 2.5: Path Relocation. Graphik nach Kindervater und Savelsbergh [1997]

2.3.4.4 Node Exchange und Path Exchange

Bei der Anwendung des Or–opt Operators auf zwei Routen von Fahrzeugen wird je eine Folge von l_1 bzw. l_2 Ecken wechselseitig zwischen den beiden Routen ausgetauscht. KINDERVATER UND SAVELSBERGH beschreiben ein Beispiel mit $l_1, l_2 = 1$ als *Exchange*, bzw. ein Beispiel mit $l_1, l_2 > 1$ als *Exchange of two Paths*. Der Fall $l_1 = 1 \wedge l_2 > 1$ bzw. $l_1 > 1 \wedge l_2 = 1$ wird nicht behandelt. In dieser Arbeit wird der wechselseitige Austausch als *Node Exchange* bezeichnet, wenn $l_1, l_2 = 1$, ansonsten wird die Bezeichnung *Path Exchange* verwendet. In Abbildung 2.6 und 2.7 sind die beiden Operationen dargestellt.

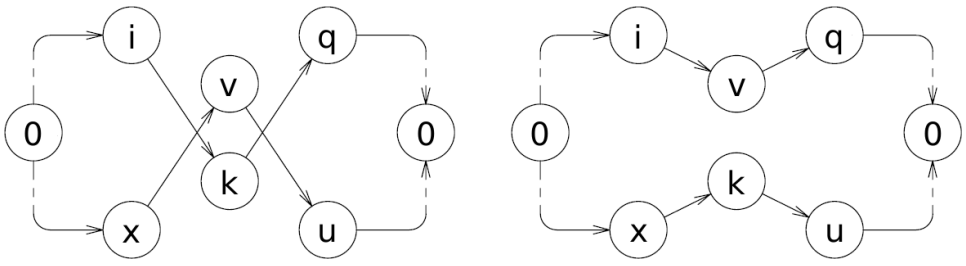

Abbildung 2.6: Node Exchange. Graphik nach Kindervater und Savelsbergh [1997]

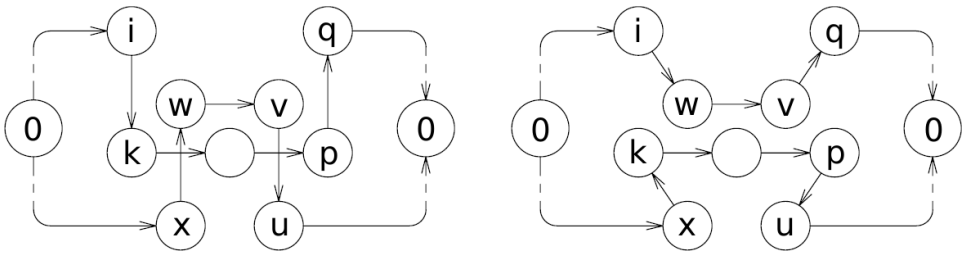

Abbildung 2.7: Path Exchange. Graphik nach Kindervater und Savelsbergh [1997]

2.3.4.5 Crossover

Beim Crossover werden bei zwei Routen jeweils eine Kante entfernt und die vier enstehenden Teilrouten "über Kreuz" verbunden. Seien $(0, \ldots, i, k, \ldots, 0)$ und $(0, \ldots, p, q, \ldots, 0)$ zwei Routen. Die Kanten (i, k) und (p, q) werden entfernt und die dadurch enstehenden Teilrouten werden durch die Kanten (i, q) und (p, k) zu zwei neuen Routen verbunden. Die Operation ist in Abbildung 2.8 dargestellt. Werden die erste Kante der einen Route sowie die letzte Kante der anderen Route entfernt, werden beide Routen zu einer Route zusammengefasst. Abbildung 2.9 zeigt die Anwendung des Crossover Operators mit anschliessender Anwendung des 2–opt Operators.

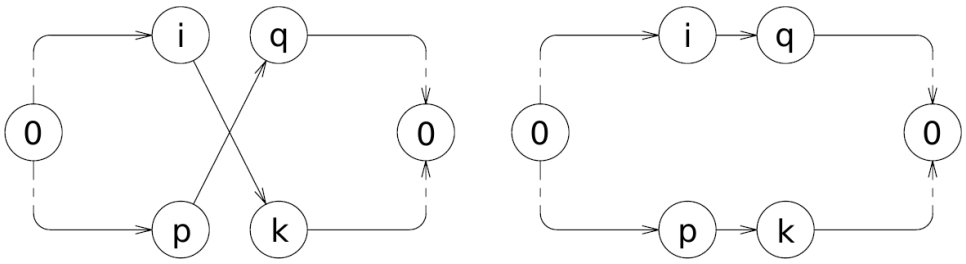

Abbildung 2.8: Crossover. Graphik nach Kindervater und Savelsbergh [1997]

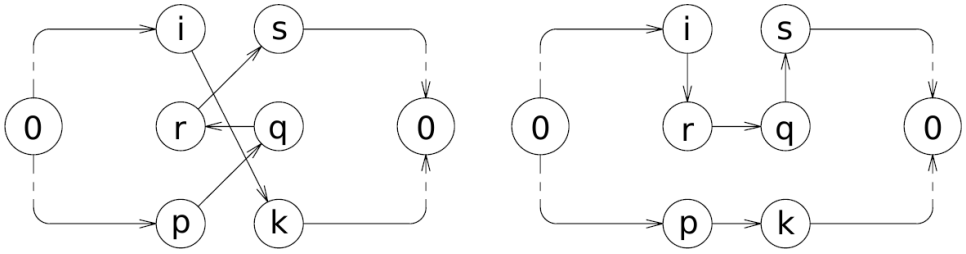

Abbildung 2.9: Crossover mit anschliessendem 2-opt. Graphik nach Kindervater und Savelsbergh [1997]

2.3.4.6 Abschließende Bemerkung zu den Modifikationsoperatoren

In der wissenschaftlichen Literatur finden sich nur Beschreibungen von Modifikationsoperatoren für das CVRP, d. h. Operatoren, die Requests, denen jeweils nur eine Ecke des Graphen zugeordnet ist, innerhalb einer Route oder zwischen Routen austauschen. Bei der Modifiktion von Pickup und Delivery Problemen (PDP) ist zu beachten, dass einem Request immer zwei Ecken zugeordnet sind. Die Abänderung der hier vorgestellten Modifikationsoperatoren zur Anwendung auf PDP ist jedoch sofort einsichtig. Es muss nur zusätzlich zu den Bedingungen des CVRP sichergestellt werden, dass die zwei Ecken, die einem PDP–Request zugeordnet sind, sich in der Route des gleichen Fahrzeugs befinden.

2.3.5 Metaheuristische Verfahren

Metaheuristische Verfahren durchsuchen iterativ, meist ausgehend von einer Startlösung, den Such– bzw. Lösungsraum, indem durch Modifikation der aktuellen Lösung bzw. einer Anzahl von aktuellen Lösungen Nachbarschaften der aktuellen Lösung bzw. Lösungen evaluiert werden. Die evaluierten Lösungen werden dabei anhand einer Bewertungsfunktion (engl. *objective funktion*) bewertet und ersetzen gegebenfalls die aktuelle Lösung bzw. eine Anzahl von aktuellen Lösungen. Dabei wird versucht, über den Suchprozess hinweg den Wert der Bewertungsfunktion zu minimieren bzw. zu maximieren. Anschaulich gesagt, “bewegt” sich dabei der Algorithmus durch den Such– bzw. Lösungsraum. Ziel der Suche sind dabei die Optima des Lösungsraums, d. h. die Lösungen, für die die Bewertungsfunktion bei der Funktionsminimierung den minimalen Wert liefert. Bei einem TSP wäre dies die Länge der kürzesten möglichen Tour. Der Fall der Funktionsmaximierung kann dazu analog angesehen werden, ist hier aber nicht von Interesse. Ein *lokales Optimum* ist eine Lösung, in dessen weiterer Umgebung sich keine bessere Lösung befindet. Ein *globales Optimum* ist eine Lösung, für die die Bewertungsfunktion den minimalen Wert liefert. Abhängig vom Problem können ein globales Optimum (monomodales Problem) oder mehrere globale Optima (mulitmodales Problem) existieren.

Metaheuristiken bauen bei ihrer Suche im Such– oder Lösungsraum auf zwei sich diametral gegenüberliegenden Konzepten auf: *Intensifikation* und *Diversifikation*. Als Intensifikation wird das Durchsuchen einer Nachbarschaft nach der besten Lösung dieser Nachbarschaft aufgefasst, wobei bessere Lösungen schlechtere Lösungen ersetzen. Dies birgt jedoch die hohe Gefahr, dass sich der Algorithmus in einem lokalen Optimum fängt. Dieser Gefahr versucht man durch die Diversifikation zu begegnen. Diese hat zum Ziel, dem Algorithmus auch das Durchsuchen anderer Nachbarschaften als der aktuellen Nachbarschaft zu ermöglichen. Um dies zu erreichen, können mehrere Methoden verwendet werden. Durch entsprechende Parameter in der Bewertungsfunktion können je nach Situation auch schlechtere Lösungen

akzeptiert werden. Diese Methode findet u. a. bei Simulated Annealing Anwendung. Insbesondere wenn pro Iteration mehrere Lösungen erzeugt werden, bietet es sich an, Lösungen, deren Akzeptanz schon feststeht, zufallsgesteuert zu verändern. Die Mutation bei genetischen Algorithmen ist hierfür ein Beispiel. Eine zu hohe Diversifikation behindert allerdings die Intensifikation. Die Ausgewogenheit zwischen Intensifikation und Diversifikation ist daher ein entscheidender Faktor für die Leistungsfähigkeit einer Metaheuristik.

Bei einigen metaheuristischen Verfahren wird bei der Suche die Generierung von Lösungen, die Randbedingungen verletzen, mit einkalkuliert. Diese Lösungen werden als *ungültige Lösungen* bezeichnet. Die Verletzung der Randbedingungen wird dabei akzeptiert, da man hofft, über ungültige Lösungen letztendlich in Bereiche des Lösungsraum zu kommen, die sehr gute Lösungen bieten. Diese Vorgehensweise wird insbesondere bei Problemen mit sehr vielen oder sehr strengen Randbedingungen wie z. B. engen Zeitfenster angewendet.

Die Literatur zu metaheuristischen Verfahren, die zur Lösung von VRP eingesetzt werden, ist unüberschaubar. Im folgenden werden die am meisten im VRP Bereich verwendeten Metaheuristiken vorgestellt. Die Angaben zur Performanz der Algorithmen wurden, soweit nicht anders angegeben, der Übersicht von GENDREAU U. A., Gendreau u. a. [2001] entnommen.

2.3.5.1 Simulated Annealing

Simulated Annealing wurde 1983 von KIRKPATRICK U. A. eingeführt [vgl. Kirkpatrick u. a., 1983] und baut auf den Arbeiten von METROPOLIS U. A. auf [vgl. Metropolis u. a., 1953]. Der Begriff “Annealing” bedeutet “ausglühen” bzw. “ausschmelzen”. Damit wird die kontrollierte, langsame Absenkung der Temperatur bei der Verarbeitung von kristallinen Werkstoffen wie z. B. Metallen bezeichnet, mit dem Ziel, dass das resultierende Werkstück einen möglichst fehlerfreien Kristall darstellt. Das Verfahren beruht auf einer der Grundlagen der statistischen Mechanik, der Gibbs–Boltzmann–Verteilung. Diese beschreibt die Wahrscheinlichkeit der Konfiguration $x \in X$ eines Systems A bei einer bestimmten Temperatur, wobei das System A sich im thermischen Gleichgewicht mit einem das System A vollständig umschliessenden System B befindet und ist durch folgende Formel gegeben:

$$P(x) = \frac{1}{Z(T)} e^{-\frac{E_X}{k_B T}}, \quad x \in X \tag{2.1}$$

wobei

X die Menge aller möglichen Konfigurationen
E_x Energieniveau der Konfiguration x
$k_B = 1.38056 \cdot 10^{-23} \frac{J}{K}$ (Boltzmannkonstante) I
Zustandsumme $Z(T) = \sum_i e^{-\frac{E_{x_i}}{k_B T}}$

Diese Formel besagt vereinfacht, dass mit zunehmender Verringerung der Temperatur die Wahrscheinlichkeit steigt, dass das System A eine Konfiguration mit niedrigerem Energieniveau annimmt. Beim Simulated Annealing wird dies angewendet, indem man das zu optimierende Problem als System ansieht und Lösungen mit Konfigurationen und den Funktionswert der Bewertungsfunktion mit dem Energieniveau der jeweiligen Konfigurationen gleichsetzt. Bei der Suche im Lösungsraum entscheidet eine von METROPOLIS U. A. entwickelte Akzeptanzfunktion stochastisch, ob eine neu generierte Lösung akzeptiert oder verworfen wird. Die Akzeptanzfunktion enthält einen Parameter T, der die "Temperatur" des Systems darstellt und die Akzeptanz von Konfigurationen beeinflusst. Wird ausgehend von der aktuellen Lösung x_i eine neue Lösung x_{i+1} erzeugt, werden für beide Lösungen anhand der Bewertungsfunktion die Funktionswerte E_{x_i} und $E_{x_{i+1}}$ berechnet und die Energiedifferenz Δ_E berechnet, indem der Wert von E_{x_i} vom Wert von $E_{x_{i+1}}$ abgezogen wird. Bei $\Delta_E < 0$ wird die neue Lösung akzeptiert. Bei $\Delta_E > 0$ ist die Wahrscheinlichkeit der Akzeptanz durch die folgende Formel bestimmt:

$$P_{accept(x)} = e^{-\frac{\Delta_E}{T}} \tag{2.2}$$

Eine Zufallszahl ϵ aus dem Intervall $[0; 1]$ entscheidet über die Akzeptanz der Lösung x_{i+1} Bei $P_{accept(x)} \leq \epsilon$ wird x_{i+1} akzeptiert, andernfalls wird x_{i+1} verworfen.

Bei einer Temperaturänderung ist das "thermische Gleichgewicht" des Optimierungsproblems nur durch eine Reihe von Konfigurationsübergängen erreichbar (die Kantengewichte einer Lösung des VRP sollten einer bestimmten Verteilung entsprechen). Dies wird durch Anwendung des Metropolis–Algorithmus erreicht [vgl. Metropolis u. a., 1953, S.1087/2,3]. Dabei werden bei konstanter Temperatur T "genügend viele" Konfigurationsübergange durchgeführt. METROPOLIS U. A. zeigen, dass sich bei einer genügend hohen Anzahl von Konfigurationsübergängen das System entsprechend der Gibbs–Boltzmann–Verteilung verhält, d. h. Konfigurationen treten mit den für sie nach der Gibbs–Boltzmann–Verteilung bestimmten Wahrscheinlichkeit auf. Dabei wird angemerkt, dass der angeführte Beweis nicht zeigt, wieviele Konfigurationsübergänge stattfinden müssen [vgl. Metropolis u. a., 1953, S.1087/,3]. Beim Simulated Annealing wird ausgehend von einer Startlösung und einer hohen Temperatur, bei der fast alle Lösungen akzeptiert werden, die Temperatur iterativ abgesenkt und nach

jeder Absenkung eine Reihe von Konfigurationsübergängen durchgeführt. Damit nähert sich das System dem Energieminimum.

KIRKPATRICK U. A. nennen die Kombination aus der Geschwindigkeit, mit der die Temperatur abgesenkt wird, und die Anzahl der Konfigurationsänderungen nach der Temperatursenkung *annealing schedule*. Für die Absenkung der Temperatur geben sie das Beispiel einer exponentieller Absenkung $T_i = (T_1/T_0)^i T_0$, wobei mit T_0 die Starttemperatur, T_1 die Temperatur nach der ersten Absenkung und mit Index i die Runde der Iteration bezeichent wird, wobei $T_0 = 10$ und $T_0/T_1 = 0.9$ [vgl. Metropolis u. a., 1953, S.1087/5]. Das annealing schedule wird auch "cooling schedule" genannt und bestimmt maßgeblich die Geschwindigkeit des Algorithmus sowie die Qualität des Endergebnisses.

GENDREAU U. A. berichten von drei Arbeiten, in denen Simulated Annealing zur Optimierung von VRP implementiert wurde. Die Ergebnisse der Implementation von ROBUSTE U. A. sind nicht überragend. Ein Vergleich zu anderen Arbeiten ist jedoch aufgrund der verwendeten Instanzen nicht möglich und auch insofern nicht sinnvoll, als Robuste u. a. die Perfomance ihrer Implementation mit der Leistungsfähigkeit von Menschen beim Lösen von TSP und VRP vergleichen. Die Arbeit kann als eine frühe Studie zum interaktiven Lösen von VRP gesehen werden [vgl. Robusté u. a., 1990]. Eine weitere von GENDREAU U. A. benannte Implementation von Simulated Annealing brachte ebenfalls keine überragende Ergebnisse [vgl. Gendreau u. a., 2001, S.2]. Einzig die Implementation OSMAN erbrachte gute Resultate, jedoch bei eher langen Laufzeiten. Dieser Algorithmus ist gegenüber dem Standard–Simulated Annealing um einiges komplexer, die Akzeptanz–Funktion sowie das cooling schedule weichen vom Standard–Simulated Annealing ab [Osman [1993], zit. n. Gendreau u. a., 2001, S.3].

Diese Ausführungen zeigen, dass die Anwendung von Simulated Annealing auf das VRP eher mit Skepsis zu betrachten ist. Allerdings zeigt sich, dass andere Heuristiken von einer Hybridisierung mit Simulated Annealing profitieren können. Einige der zur Zeit am leistungsfähigsten Algorithmen zur Lösung von VRP benutzen Simulated Annealing als Meta–Strategie oder in anderer Form. Ein Beispiel dafür ist das in einem der folgenden Kapitel vorgestellte *Adaptive Large Neighbourhood Search* (ALNS).

2.3.5.2 Tabu–Suche

Bei der Durchsicht bestehender Literatur zu Lösungsverfahren von VRP dominieren unter den leistungsfähigeren Algorithmen die Tabu–Suche und deren Varianten. Eine Kurzeinführung in die Tabu–Suche findet sich bei LAPORTE U. A. und GENDREAU U. A.. Bei der Tabu–Suche werden wie beim Simulated Annealing iterativ Lösungen aus der Nachbarschaft untersucht. Es wird jedoch bei jedem Schritt nur die beste Lösung aus der Nachbarschaft akzeptiert. Die beste Lösung ist dabei nicht unbedingt diejenige, für die die Bewertungsfunktion

den besten Wert liefert [vgl. Laporte u. a., 2000, S.6]. Während der Suche wird eine Liste der jeweils letzten n Lösungen, die evaluiert wurden, mitgeführt. Diese n Lösungen sind für eine erneute Evaluation “tabu”. Durch diesen Mechanismus sollen Zyklen bei der Suche vermieden werden. Um einen hohen zeitlichen Aufwand sowie Speicherbedarf zu umgehen, werden anstatt der Lösung nur bestimmte Eigenschaften der Lösungen abgespeichert. Die genannten Bestandteile stellen die Basis–Version der Tabu–Suche dar, von der mehrere Erweiterungen mit Mechanismen zur Intensifikation und Diversifikation existieren [vgl. Laporte u. a., 2000, S.6] und [Gendreau u. a., 2001, S.5,6]. Die in bei LAPORTE U. A. und GENDREAU U. A. erwähnten performanten Varianten der Tabu–Suche werden in den folgenden Abschnitten repräsentiert.

In der Implementation von OSMAN werden Lösungen aus der Nachbarschaft durch den k–opt Operator mit $k = 2$ erzeugt. Zusätzlich werden Ecken zwischen Routen verschoben bzw. ausgetauscht. Von diesem Algorithmus existieren zwei Versionen: in der Version BA (best–admissible) wird die gesamte Nachbarschaft durchsucht und die beste gültige Lösung, die nicht tabu ist, wird ausgewählt. In der Version FBA (first–best–admissible) wird die erste gültige Lösung, die die bestehende Lösung verbessert ausgewählt. Besteht in der Nachbarschaft keine bessere gültige Lösung als die gegenwärtige Lösung, wird die beste gültige Lösung aus der Nachbarschaft ausgewählt [Osman:1993, zit. n. Gendreau u. a., 2001, S.6].

Eine von GENDREAU U. A. entwickelte Implementation, Taburoute genannt, ist um einiges aufwendiger als Osmans TS Algorithmus. Die Nachbarschaftsstruktur ist durch alle Lösungen definiert, die durch Verschieben einer Ecke in eine neue Route entstehen können, wobei die neue Route p nächste Ecken aufweisen muss. Der Operator, GENI (*GENeralized Insertion procedure*), von Gendreau, Hertz und Laporte für das TSP entwickelt, ist in [Gendreau:1992, zit. n. Gendreau u. a., 2001] detailliert beschrieben. Bei Taburoute werden auch Lösungen zugelassen, die nicht gültig sind. die Bewertungsfunktion enthält dazu zwei Strafwerte, einen für Kapazitäts– und einen für Zeitlimit–Überschreitung. Diese Strafwerte werden je nach Situation angepasst. Sind zehn aufeinander folgende Lösungen bezüglich eines Limits gültig, wird der entsprechende Starfwert halbiert, sind zehn aufeinander folgende Lösungen bezüglich eines Limits ungültig, wird der entsprechende Strafwert verdoppelt. Während des Suchprozesses werden die Routen durch einen sogenannten US Operator (*Unstringing and Stringing*) optimiert [Gendreau:1992, zit. n. Gendreau u. a., 2001, S. 6–7]. Anstatt einer Tabuliste werden Tabutags benutzt. Wenn eine Ecke aus einer Route herausgenommen wird, ist das Wiedereinfügen dieser Ecke in die Route für eine aus dem Intervall $[5, 10]$ zufallsbestimmte Anzahl von Iterationen verboten. Ecken, die öfter bewegt werden, werden ebenfalls mit einem Strafwert versehen, so das Ecken, die wenig bewegt wurden, favorisiert werden. Am Anfang der Optimierung werden eine Menge von Startlösungen erzeugt und diese jeweils für einige Runden optimiert. Die beste Lösung wird als Startlösung für die weitere Optimierung herangezogen [Gendreau:1994, zit. n. Gendreau u. a., 2001, S. 7].

Von TAILLARD wird ein Algorithmus beschrieben, der Eigenschaften von Osman's Tabu–Suche und Taburoute vereinigt. Die Nachbarschaft ist durch den k–opt Operator wie bei Osman definiert. Von Taburoute sind die Tabutags und das Belegen oft bewegter Ecken durch Strafwerte übernommen. Eine charakteristische Eigenschaft des Algorithmus von Taillard ist die Unterteilung des VRP in Teilprobleme durch Unterteilung des dem VRP zugrundeliegenden Graphen in Zonen. Bei planaren Graphen erfolgt eine Unterteilung in vom Depot ausgehende Sektoren (ähnlich wie beim Sweep-Algorithmus). Die Sektoren wiederum sind in konzentrische Zonen unterteilt. Jedes Teilproblem kann unabhänig für sich gelöst werden, allerdings müssen periodisch Ecken zu benachbarten Zonen zugeteilt werden. Dieses Vorgehen macht Sinn, wenn das Depot "in der Mitte" liegt und die übrigen Ecken annähernd gleichverteilt sind. Durch die Aufteilung des Problems in Teilprobleme ist dieses Verfahren gut für eine verteilte Anwendung, bei der Teilprobleme jeweils einem Prozessor zugeordnet werden, geeignet [Gendreau u. a. [2001], zit. n. Taillard, 1993].

Das Konzept des *Adaptive Memory* wird von GENDREAU U. A. als eine der interssantesten Entwicklungen aus dem Bereich der Tabu–Suche innerhalb der letzten Jahre bezeichnet. Das Konzept wurde von ROCHARD UND TAILLARD eingeführt. Ein Adaptive Memory ist ein Pool von guten Lösungen, der während der Optimierung aktualisiert wird. Aus diesem Pool werden periodisch neue Lösungen generiert, indem aus mehreren Lösungen des Pools jeweils eine Teillösung (Route) entnommen wird und die entsprechenden Teillösungen zusammengefügt werden. Da bei den entnommenen Lösungen Ecken in zwei oder mehr Routen bzw. in keiner Route enthalten sein können, entstehen oft Teillösungen, die durch eine Konstruktionsheuristik vervollständigt werden müssen [Rochat:1995, zit. n. Gendreau u. a., 2001].

Ein relativ neues und vielversprechendes Konzept ist das *Granular Tabu Search* (GTS) von TOTH UND VIGO. Es beruht auf der Beobachtung, dass höhergewichtete Kanten eine geringere Wahrscheinlichkeit besitzen, Teil einer qualitativ hochwertigen Lösung zu sein. Beim GTS werden deshalb alle Kanten, die ein höheres Gewicht als das sogenannte *granularity threshold* besitzen, entfernt. Hiervon ausgenommen sind die Kanten, die das Depot mit einer anderen Ecke verbinden. Toth und Vigo schlagen für das *Granularity Threshold* einen wert $v = \beta\overline{c}$ vor, wobei der Ausdünnungsparameter (*Sparsification Parameter*) β aus dem Intervall $[1.0, 2.0]$ stammt und $\overline{c}$ das durch eine schnelle Heuristik bestimmte durchschnittliche Kantengewicht darstellt. Bei $\beta in [1.0, 2.0]$ liegen nach der Ausdünnung die restlichen Kanten im Bereich von 10% – 20% aller Kanten. Während des Optimierungsvorgangs wird β dynamisch angepasst. Treten während einer bestimmten Anzahl von Iterationen keine Verbesserungen der aktuellen Lösung auf, wird β etwas erhöht und danach langsam wieder auf seinen urspünglichen Wert abgesenkt. Die Suche im Lösungsraum wird durch Tausch von Kanten innerhalb einer Route oder durch Tausch von Kanten zwischen Routen realisiert. Eine von Toth und Vigo implementierte Variante des GTS enthält einige Features von Taburoute [Toth:2003, zit. n. Gendreau u. a., 2001].

2.3.5.3 Ant Colony Optimization

Die Familie der Ant Colony Optimization Algorithmen (ACO) hat sich aus den von Colorni, Dorigo und Maniezzo in den frühen 1990er Jahren entwickelten Ant Systems (AS) entwickelt. Vorbild bei der Entwicklung war das emergente Verhalten von Ameisenkolonien bei der Futtersuche. Das Verhalten einer einzelnen Ameise ist dabei zunächst nicht zielgerichtet, die Ameise wandert mehr oder weniger zufällig durch die Gegend. Während der Fortbewegung wird der zurückgelegte Pfad mit Pheromonen markiert. Stößt eine Ameiese auf einen mit Pheromonen markierten Pfad, entscheidet sie sich mit höherer Wahrscheinlichkeit dafür, diesem Pfad zu folgen als den Weg durch zufälliges Herumzuwandern fortzusetzen. Die einem mit Pheromonen markierten Pfad folgenden Ameisen markieren diesen ebenfalls mit Pheromonen, die Pheromonkonzentration des Pfades steigt dadurch an. Die abgelegten Pheromone verwittern mit der Zeit. Transportiert eine Anzahl von Ameisen Futter von einer Futterquelle aus auf zwei verschieden langen Wegen zum Bau, ist zunächst die Auswahlwahrscheinlichkeit für beide Wege gleich hoch. Dadurch befinden sich auf dem längeren Weg weniger Ameisen pro Zeiteinheit und Wegeinheit. Die Pheromonkonzentration des längeren Pfades sinkt dadurch gegenüber der des kürzeren Pfades ab. Dieser Effekt ist selbstverstärkend, was dazu führt, dass der kürzere Pfad immer mehr favorisiert wird.

COLORNI U. A. beschreiben die Funktionsweise der AS–Algorithmen anhand des TSP mit euklidischen Distanzen [vgl. Colorni u. a., 1991, S.3]. Dabei werden drei Varianten, ANT–Density, ANT–Quantity und ANT–Cycle aufgeführt. Colorni u. a. verwenden dabei die Bezeichnung *ant–algorithms* [vgl. Colorni u. a., 1991, S.3], der Begriff AS wurde erst später allgemein verwendet, um AS von den daraus später entwickelten ACO Algorithmen wie z. B. das *Ant Colony System* (ACS) abzugrenzen [vgl. u. a. Dorigo und Stützle, 2003, S.12] sowie [Coltorti und Rizzoli, 2007, S.3]. Im folgenden wird das Grundschema, das für alle AS–Varianten gleich ist, dargestellt. Danach wird auf die Eigenheiten jeder Variante eingegangen.

Beim TSP ist diejenige Kantenfolge durch alle Ecken eines Graphen zu finden, deren Summe der Kantengewichte minimal ist, wobei die Anzahl der Ecken des Graphen mit n bezeicnet wird. Bei der Anwendung des AS auf das TSP bezeichnet $b_i(t)$ mit $i = 1, \ldots, n$ die Anzahl der Ameisen, die sich zum Zeitpunkt t auf der Ecke i befinden. Die Anzahl m aller Ameisen ist durch

$$m = \sum_{i=1}^{n} b_i(t) \tag{2.3}$$

definiert. Die Pheromonkonzentration einer Kante d_{ij} ist für einen Zeitpunkt t durch den Wert $\tau_{ij}(t)$ bestimmt, wobei $\tau_{ij}(t)$ für $t = 0$ auf einen beliebigen, geringen Wert gesetzt wird. Der Anfangswert wird von COLORNI U. A. nicht genau angegeben (“in our experiments a very

low value on every path$_{ij}$") [Colorni u. a., 1991, S.4]. Die Pheromonkonzentration $\tau_{ij}(t)$ wird iterativ durch die folgende Formel aktualisiert:

$$\tau_{ij}(t+1) = \rho \cdot \tau_{ij}(t) + \Delta\tau_{ij}(t,\, t+1) \tag{2.4}$$

wobei ρ den Zerfallsfaktor der Pheromonen und $\Delta\tau_{ij}(t,\, t+1)$ den Zuwachs an Pheromonkonzentration darstellt. Dieser ist durch die Formel

$$\Delta\tau_{ij}(t,\, t+1) = \sum_{k=1}^{m} \Delta\tau_{ij}^{k}(t,\, t+1) \tag{2.5}$$

definiert, wobei m die Anzahl der Ameisen und $\Delta\tau_{ij}^{k}(t,\, t+1)$ die von der k–ten Ameise auf der Kante d_ij während des Zeitintervalls $[t,\, t+1]$ abgegebene Pheromonmenge bezeichnet. Diese Pheromonmenge wird für jede der drei AS–Varianten unterschiedlich berechnet. In Berechnung der Auswahlwahrscheinlichkeit einer Kante d_{ij} wird noch die sogenannte *Sichtbarkeit* η_{ij} einer Kante mit einbezogen, diese beträgt $\eta_{ij} = 1/w_{ij}$, wobei w_{ij} das Gewicht der Kante bezeichnet. Die Wahrscheinlichkeit für die Auswahl einer Kante d_{ij} durch die Ameise k ist durch

$$P_{ij}^{k}(t) = \begin{cases} \dfrac{[\tau_{ij}(t)]^{\alpha} \cdot [\eta_{ij}(t)]^{\beta}}{\sum_{l \in \mathcal{N}_i^k} [\tau_{il}(t)]^{\alpha} \cdot [\eta_{il}(t)]^{\beta}} & \text{wenn } j \in \mathcal{N}_i^k \\ 0 & \text{sonst} \end{cases} \tag{2.6}$$

definiert, wobei $\mathcal{N}_i^k$ die Menge der Ecken ist, die die Ameise noch nicht besucht hat. Durch die Parameter α und β wird der Einfluss der Pheromonkonzentration auf einer Kante bzw. der Sichtbarkeit der Kante festgelegt.

Im ANT–Quantity Modell ist der Zuwachs an Pheromonkonzentration durch

$$\Delta\tau_{ij}^{k}(t,\, t+1) = \begin{cases} \frac{Q_1}{w_{ij}} & \text{wenn die } k\text{–te Ameise im Intervall } [t,\, t+1] \text{ Kante } d_{ij} \text{ benutzt} \\ 0 & \text{sonst} \end{cases} \tag{2.7}$$

bestimmt, wobei Q_1 eine konstante Pheromonmenge ist. Im ANT–Density Modell beträgt der Zuwachs an Pheromonkonzentration

$$\Delta\tau_{ij}^{k}(t,\, t+1) = \begin{cases} Q_2 & \text{wenn die } k\text{–te Ameise im Intervall } [t,\, t+1] \text{ Kante } d_{ij} \text{ benutzt} \\ 0 & \text{sonst} \end{cases} \tag{2.8}$$

wobei Q_2 eine konstante Pheromonmenge pro Kantengewichtseinheit darstellt.

Der Algorithmus der ANT–Quantity und ANT–Density Variante ist im Wesentlichen gleich und wird im folgenden dargestellt.

1. Bei der Initialisierung wird jede Ecke des Graphen mit einer Anzahl von Ameisen besetzt und t wird auf 0 gesetzt. Für jede im Graph enthaltene Kante d_{ij} wird eine initiale Pheromonkonzentration $\tau_{ij}(t)$ gesetzt und es gilt $\Delta\tau_{ij}(t,\, t+1) = 0$.
2. Jede Ameise wählt von ihrer aktuellen Postion ausgehend stochastisch die nächste zu besuchende Ecke anhand der in Formel 2.10 definierten Kantenübergangswahrscheinlichkeit. Für jede von einer Ameise ausgewählte Kante wird der Zuwachs an Pheromonkonzentration nach Formel 2.8 bzw. 2.9 neu berechnet.
3. Die Pheromonkonzentration aller Kanten wird neu berechnet. Haben die Ameisen alle Ecken des Graphen durchlaufen, weiter mit Schritt 3, sonst weiter mit Schritt 2.
4. Die bisher kürzeste gefundene Tour wird abgespeichert. t wird inkrementiert $(t = t + 1)$. Für jede Kante des Graphen gilt $\Delta\tau_{ij}(t,\, t+1) = 0$. Wenn die Abbruchbedingung erfüllt ist, wird terminiert, sonst weiter mit Schritt 2.

Als Abbruchbedingung geben COLORNI U. A. das Erreichen einer bestimmten Anzahl von Durchläufen an [vgl. Colorni u. a., 1991, S.5]. Alternativ dazu könnte das Erreichen einer bestimmten Anzahl von Durchläufen ohne Verbesserung der bisher besten gefundenen Tour als Abbruchbedingung definiert werden.

Der ANT–Cycle Algorithmus unterscheidet sich vom ANT–Quantity und ANT–Density Algorithmus darin, dass $\Delta\tau_{ij}^{k}$ und die Pheromonkonzentration aller Kanten erst dann neu berechnet werden, wenn alle Ameisen eine TSP–Tour beeendet haben. $\Delta\tau_{ij}^{k}(t,\, t+n)$ ist durch die Formel

$$\Delta\tau_{ij}^{k}(t,\, t+n) = \begin{cases} \frac{Q_3}{L^k} & \text{wenn die TSP-Tour der } k\text{–ten Ameise die Kante } d_{ij} \text{ enthält} \\ 0 & \text{sonst} \end{cases} \tag{2.9}$$

definiert, wobei L^k die Summe der Kantengewichte aller Kanten der TSP-Tour der Ameise k ist.

DORIGO UND STÜTZLE geben an, dass für eine sehr hohe Performanz von ACO–Algorithmen bei einigen Anwendungen die Miteinbeziehung von lokalen Verbesserungsheuristiken von entscheidender Bedeutung ist. Trotzdem werden auch bei Anwendungen, bei denen die Verwendung von lokalen Verbesserungsheuristiken nicht möglich ist, sehr gute Ergebnisse erzielt [vgl. Dorigo und Stützle, 2003, S.27–28].

Einige erfolgreiche Anwendungen eines ACO–Algorithmus auf VRP basieren auf dem von Gambardella, Taillard und Agazzi entwickelten *Multiple Ant Colony System for Vehicle Routing with Time Windows* (MACS-VRPTW), dass in Gambardella u. a. [1999] vorgestellt wird. MACS-VRPTW basiert auf dem Ant Colony System Algorithmus (ACS), die folgende Beschreibung des ACS ist Gambardella u. a. [1999] entnommen.

ACS weist gegenüber den AS Algorithmen einige Unterschiede auf. Die Aktualisierung der Pheromonkonzentrationen der Kanten geschieht sowohl während der Konstruktion der TSP-Touren aller Ameisen als auch bei den fertigen TSP–Touren. GAMARDELLA U. A. bezeichnen dies als *local update* und *global update*. Für das globale Update wird jeoch nur die beste erzeugte Tour verwendet [vgl. Gambardella u. a., 1999, S.5].

Das Auswahlverfahren, mit dem entschieden wird, welche Kante als nächstes durch eine Ameise beschritten wird, ist zweistufig. Mit einer Wahrscheinlichkeit von q_0 wird diejenige Kante ausgewählt, durch die $\tau_{ij}[\eta_{ij}]^\beta$ maximiert wird. Mit einer Wahrscheinlichkeit von $1 - q_0$ wird die Auswahlwahrscheinlichkeit einer Kante d_ij durch die Formel

$$P_{ij} = \begin{cases} \dfrac{\tau_{ij} \cdot [\eta_{ij}(t)]^\beta}{\sum_{l \in \mathcal{N}_i^k} \tau_{il} \cdot [\eta_{il}(t)]^\beta} & \text{wenn } j \in \mathcal{N}_i^k \\ 0 & \text{sonst} \end{cases} \tag{2.10}$$

bestimmt. Die Formel entspricht Formel 2.10 mit $\alpha = 1$. Das lokale Update der Pheromonkonzentrationen geschieht nach der Formel

$$\tau_{ij} = (1 - \rho) \cdot \tau_{ij} + \rho \cdot \tau_0 \tag{2.11}$$

wobei $0 \leq \rho \leq 1$ und τ_0 die initiale Pheromonkonzentration ist. GAMBARDELLA U. A. empfehlen für τ_0 den Wert $\tau_0 = (n \cdot J_\Psi^h)^-1$, dabei ist n die Anzahl der Ecken des Graphen und J_Ψ^h ist die Länge des mit dem Nearest–Neighbour–Algorithmus erzeugten Hamiltonkreises [vgl. Gambardella u. a., 1999, S.6]. Das globale Update ist durch

$$\tau_{ij} = (1 - \rho) \cdot \tau_{ij} + \frac{\rho}{J_\Psi^{gb}} \qquad \forall (i,j) \in \Psi^{gb} \tag{2.12}$$

definiert, wobei J_{Ψ}^{gb} die Summe der Kantengewichte der besten Lösung Ψ^{gb} seit Beginn des Optimierungsprozesses ist.

Die folgende Beschreibung des MACS-VRPTW stammt ebenfalls von GAMBARDELLA U. A.. Beim MACS–VRPTW werden zwei ACS parallel eingesetzt, wobei ein ACS zur Minimierung der Anzahl der Fahrzeuge (ACS–VEI) und ein ACS zur Minimierung der Tourlängen (ACS-TIME) dient. Beide ACS arbeiten mit jeweils eigenen Pheromonkonzentrationen, benutzen aber beide gemeinsam Ψ^{gb}, die seit Anfang des Optimierungsprozesses beste erzeugte Lösung. Als Startlösung wird eine (gültige) Lösung mit dem Nearest–Neighbour Algorithmus erzeugt. ACS-VEI versucht, eine gültige Lösung zu erzeugen, die ein Fahrzeug weniger als Ψ^{gb} aufweist. ACS-TIME optimiert den Zeitbedarf von Ψ^{gb}. Gelingt es ACS-VEI, eine gültige Lösung mit einem Fahrzeug weniger als Ψ^{gb} zu erzeugen, werden ACS-VEI und ACS-TIME terminiert und jeweils ein neu initialisiertes ACS–VEI und ACS–TIME mit der neuen, um ein Fahrzeug reduzierten Lösung gestartet [vgl. Gambardella u. a., 1999, S.6,7].

COLTORTI UND RIZZOLI erwähnen ein auf MACS-VRPTW basierendes System zur Optimierung von realen VRP, das kommerzielle Produkt ANTROUTE.

Ein Szenario betrifft eine Logistikfirma in Italien, bei der täglich 1000 – 1500 Fahrzeuge mit einer Tourdauer von einem bis drei Tagen eingesetzt werden. Das betreffende Optimierungsproblem entspricht einem PDPTW, wobei wiederum drei verschieden Arten von Fahrzeugen mit jeweils unterschiedlicher Kapazität zum Einsatz kommen. Bei einem zweiwöchigen Vergleichstest zwischen menschlichen Planern und ANTROUTE konnten durch ANTROUTE 2,63% an Touren (absolut 10,7 Touren) sowie 1,32% an Kilometern (absolut 1818,2 Km) eingespart werden, die insgesamt erreichte Effizienzsteigerung wird mit 4,19% angegeben [vgl. Coltorti und Rizzoli, 2007, S.3–5].

Ein Beispiel für ein reales DSVRP findet sich ebenfalls bei COLTORTI UND RIZZOLI. Für einen Öl-Distributor mit einer Fahrzeugflotte von 10 Fahrzeugen sind die Routen bei einem Planungshorizont von acht Stunden zu optimieren. Bei mehreren Tests wurde der Planungshorizont in fünf bis 200 Zeitscheiben von 5760 Sekunden bis 144 Sekunden unterteilt. Die Ergebnisse der Tests werden jeweils als Tourdauer angegeben, wobei das beste Ergebnis mit 9744 Sekunden bei 25 Zeitscheiben von je 1152 Sekunden erzielt wird. Demgegenüber stehen 12702 Sekunden bei 200 Zeitscheiben von je 144 Sekunden, sowie 11201 bei 5 Zeitscheiben von je 5760 Sekunden. Ein Vergleich mit der Leistungsfähigkeit eines menschlichen Planers wird für dieses Problem nicht angegeben [vgl. Coltorti und Rizzoli, 2007, S.7,8].

2.3.5.4 Genetische Algorithmen

Die Funktionsweise von genetischen Algorithmen ist an die Prozesse der natürlichen Evolution angelehnt. Für die Einführung des Modells werden die Arbeiten von John H. Holland

(Holland [1975]) genannt [vgl. u. a. Kennedy und Eberhart, 2001, S.137], [Whitley, 1994, S.1]. Die Grundlagen der genetischen Algorithmen sind diesen beiden Quellen entnommen. Das Grundprinzip der genetischen Algorithmen besteht darin, bei der Optimierung eines gegebenen Problems iterativ eine meist größere Menge von Lösungen zu erzeugen, wobei die in einer Iteration erzeugten Lösungen aus Teilen von Lösungen der vorhergehenden Iteration zusammengesetzt werden. Die gesamte Menge von Lösungen einer Iteration wird als *Generation* oder *Population*, die einzelnen Lösungen einer Generation werden als *Chromosom* oder auch *Individuum* bezeichnet. Die durch ein Chromosom repräsentierte Lösung ist im Chromosom durch *Gene* kodiert, d. h. die Menge aller möglichen Gene stellt das Alphabet für die Kodierung dar. Die in einem Chromosom enthaltene Anzahl von Genen ist für alle Chromosmomen innerhalb einer Anwendung eines genetischen Algorithmus gleich. Bei der Erzeugung neuer Chromosomen für die Folgegeneration werden stochastisch Chromosomen der aktuellen Generation ausgewählt, wobei diejenigen Chromosomen, die eine bessere Lösung repräsentieren, mit höherer Wahrscheinlichkeit ausgewählt werden. In einem *Rekombination* bzw. *Crossover* genannten Vorgang werden aus jeweils zwei ausgewählten Chromosomen Gene entnommen und zu Chromosomen der Folgegeneration zusammen gesetzt. Die erste Generation eines genetischen Algorithmus – auch Anfangspopulation genannt – wird meist zufallsgesteuert generiert.

Die Qualität eines Chromosoms wird als *Fitness* bezeichnet. Sowohl WHITLEY als auch KENNEDY UND EBERHARD stellen fest, dass der durch die Bewertungsfunktion bestimmte Wert oft mit der Fitness gleichgesetzt wird und betonen die Wichtigkeit der Unterscheidung zwischen Bewertungsfunktion und Fitnessfunktion. Die Fitnessfunktion wandelt den für ein bestimmtes Chromosom durch die Bewertungsfunktion bestimmten Wert in die Fitness um [vgl. Whitley, 1994, S.4] sowie [Kennedy und Eberhart, 2001, S.153–154].

Zur Berechnung der Fitness existieren verschiedene Verfahren. Beim kanonischen genetischen Algorithmus ist die Fitness durch $f_i/\overline{f}$ bestimmt, wbei f_i der Wert der Bewertungsfunktion für das Chromosom i und $\overline{f}$ der Mittelwert der Werte der Bewertungsfunktion für alle Chromosmen ist [vgl. Whitley, 1994, S.4]. Eine andere Möglichkeit ist die Berechnung der Fitness eines Chromosoms als normierten Wert [vgl. Kennedy und Eberhart, 2001, S.155–156].

Zur Auswahl von Chromosomen zur Rekombination werden ebenfalls verschiedene Verfahren benutzt. In Verbindung mit normierter Fitness wird oft die *Roulette Wheel Selection* angewendet, bei der die Wahrscheinlichkeit zur Auswahl eines Chromosoms proportional zur dessen Fitness ist. Bei der Bildung der Folgegeneration kommt bisweilen auch die *Elitist Strategy* zum Einsatz, bei der das Chromosom mit der höchsten Fitness in die Folgegeneration übernommen wird [vgl. Kennedy und Eberhart, 2001, S.156–157].

Auch für die Rekombination existieren verschiedene Verfahren, wobei die gebräuchlichsten der One–Point–Crossover, Two–Point–Crossover sowie der Uniform–Crossover sind.

Beim One–Point–Crossover wird stochastisch die Stelle bestimmt, an der beide “Eltern”–Chromosomen jeweils in einen Anfangs– und Endteil aufgetrennt werden. Der Anfangsteil des einen Eltern–Chromosoms wird jeweils mit dem Endteils des anderen Eltern–Chromosoms verbunden, wodurch zwei neue Chromosomen für die Folgegeneration erzeugt werden. Abbildung 2.10 veranschaulicht die Operation.

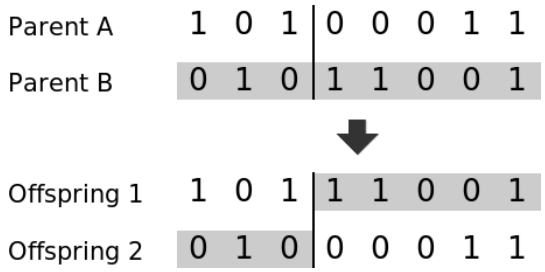

Abbildung 2.10: One–Point–Crossover

Beim Two–Point–Crossover werden zwei Schnittpunkte bestimmt, an denen die Eltern–Chromosomen jeweils in Anfangs–, Mittel– und Endteil aufgeteilt werden. Zur Erzeugung von Chromosomen der Folgegeneration wird zwischen beiden Eltern–Chromosomen der Mittelteil bzw. der Anfangs– und Endteil ausgetauscht. Abbildung 2.11 veranschaulicht die Operation.

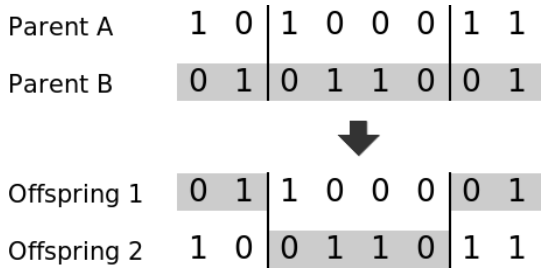

Abbildung 2.11: Two–Point–Crossover

Beim Uniform–Crossover wird für jede Stelle in den Eltern–Chromosomen stochastisch bestimmt, ob die Gene an der entsprechenden Stelle ausgetauscht werden. Durch den wechselseitigen Austausch der Gene an den bestimmten Stellen entstehen wiederum zwei Chromosomen der Folgegeneration. Der Vorgang ist in Abbildung 2.12 dargestellt.

Nach der Rekombination werden die Chromosomen der Folgegeneration mit einer geringen Wahrscheinlichkeit mutiert. Bei der Mutation eines Chromosoms werden Gene zufallsgesteuert ausgewählt und verändert. Während die Rekombination für die Intensifikation bei der Suche sorgt, dient die Mutation der Diversifikation. Der Ablauf eines einfachen genetischen Algorithmus sieht damit wie folgt aus:

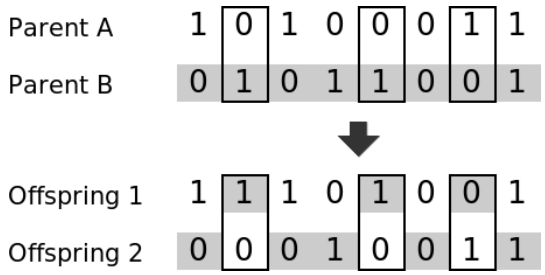

Abbildung 2.12: Uniform–Crossover

1. Initialisiere die Anfangspopulation.
2. Berechne die Fitness für jedes Chromosom.
3. Wähle die Chromosomen zur Bildung der Folgegeneration.
4. Bilde die Folgegeneration durch Rekombination von Chromosomen. Wende mit geringer Wahrscheinlichkeit einen Mutationsoperator auf die Chromosomen der Folgegeneration an.
5. Wenn die Abbruchbedingung erfüllt ist, gib das Chromosom mit der höchsten Fitness zurück und terminiere. Ansonsten weiter mit Schritt 2.

Die im folgenden beschriebenen Ansätze zur Anwendung von genetischen Algorithmen behandeln alle das CVRP bzw. CVRPTW.

Bei der Anwendung von genetischen Algorithmen auf das VRP bestehen zwei Problemfelder: die Repräsentation einer Lösung bzw. deren Kodierung als Chromosom und die Art der Crossover–Operatoren. Das erste Problem wird bisher in fast allen Lösungsansätzen mehr oder weniger gleich gehandhabt. Die einzelnen Routen einer Lösung werden als eine geordnete Liste von Ecken nacheinander zu einem Chromosom zusammengefügt, wobei die Reihenfolge der Ecken innerhalb einer Liste der Reihenfolge der Ecken in der entsprechenden Route entspricht. In einigen Modellen wird im Chromosom zwischen den Routen ein spezielles Gen zur Trennung der Routen eingefügt. DUNCAN fügt im Chromosom vor den Anfang jeder Route das Depot als Gen zur Trennung ein [vgl. Duncan, 1995, S.4]. Bei OMBUKI U. A. sind die Routen in einem Chromosom ohne ein spezielles Gen als Trennzeichen hintereinander angeordnet [vgl. Ombuki u. a., 2006, S.11]. Ombuki u. a. geben allerdings keine Hinweise darauf, wie bei der Anwendung von Chrossover und Mutation bei einem Chromosom die Erkennung der einzelnen Routen innerhalb des Chromosoms erfolgt.

Bei der eben beschriebenen Repräsentation einer Lösung durch ein Chromosom würde die Anwendung der sonst üblichen Standard–Crossover–Operatoren mit hoher Wahrscheinlichkeit dazu führen, dass in den Chromosomen der Folgegeneration einerseits bestimmte Ecken fehlen, andererseits einige Ecken doppelt in einem Chromosom enthalten sind. Die so

generierten Lösungen wären folglich mit hoher Wahrscheinlichkeit ungültig. Diesem Problem wird mit unterschiedlichen Methoden begegnet.

DUNCAN verwendet drei Operatoren mit den Bezeichnungen “Swap”–Operator, “Re–Insert”–Operator und “Relink”–Operator. Die drei Operatoren entsprechen den im Kapitel 2.3.4, Modifikationsoperatoren, besprochenen Operatoren Node–Exchange, Or–Opt und k–Opt mit $k = 2$. Bei Verletzung von Randbedingungen wird auf den Wert der Bewertungsfunktion ein “Strafwert” aufgeschlagen [vgl. Duncan, 1995, S.3–5]. Nach der Beschreibung von Duncan kann gefolgert werden, dass die Operatoren zufallsgesteuert entweder auf einzelne Routen oder auf jeweils zwei Routen angewendet werden. Der Genetische Algorithmus ist in fünf Varianten implementiert. Diese werden anhand von fünf Instanzen aus dem Christofides Benchmark mit einer von Duncan implementierten Tabu–Suche verglichen. Die Instanzen enthalten zwischen 50 und 100 Requests. Die Implementation der Tabu–Suche schneidet gegenüber der besten Variante des genetischen Algorithmus bei den Instanzen mit 50 und 75 Requests etwas besser ab und fällt bei den übrigen Instanzen gegenüber dem genetischen Algorithmus leicht zurück [vgl. Duncan, 1995, S.11].

POTVIN UND BENGIO verwenden den *Sequence–Based Crossover*, bei dem bei beiden Eltern–Chromosomen Parent A und Parent B jeweils eine Kante zufallsgesteuert entfernt wird. Der Anfang der aufgetrennten Route von Parent A wird mit dem Endteil der aufgetrennten Route von Parent B verbunden. Die dadurch entstehende Lösung enthält mit hoher Wahrscheinlichkeit Ecken doppelt oder gar nicht. In diesem Fall werden durch einen *Repair Operator* doppelt enthaltene Ecken gelöscht bzw. nicht enthaltene Ecken an die günstigste Stelle eingefügt. Beim *Route Based Crossover* wird eine Route von Parent A mit einer Route von Parent B ausgetauscht. Nach Anwendung dieses Operators muss meist ebenfalls durch nachfolgende Anwendung des Repair Operators die entstandene Lösung in eine gültige Lösung überführt werden [vgl. Potvin und Bengio, 1996, S.6–9]. Bei einer Anwendung des Algorithmus auf Instanzen des Solomon Benchmarks wird bei den Instanzen C102, C103, C106, C107, C108 das bekannte beste Optimum erreicht. Bei den Instanzen R102 liegt der Algorithmus ca. 3,8%, bei der Instanz R103 ca. 6,8% über dem besten bekannten Optimum. Für die anderen Instanzen wird kein Vergleich mit dem bekannten besten Optimum angegeben.

OMBUKI U. A. stellen den *Best Cost Route Crossover* (BCRC) Crossover–Operator vor. Bei diesem Operator wird zufallsgesteuert aus beiden Elternchromosomen jeweils eine Route entnommen und die Ecken dieser Route in das jeweils andere Elternchromosom an die jeweils günstigste Stelle eingefügt. Bevor die Ecken einer Route in ein Chromosom eingefügt werden, werden die in dieser Route enthaltenen Ecken aus dem Chromosom gelöscht, so dass in einem Chromosom jede Ecke nur einmal enthalten ist. Kann eine Ecke aufgrund von Randbedingungen nicht in eine bestehende Tour eingefügt werden, wird mit dieser Ecke eine neue Tour begonnen. Das Verfahren wird anhand des Solomon Benchmarks getestet. Bei den Klassen C1 und C2 wird das bekannte beste Optimum in den meisten Fällen erreicht

[vgl. Ombuki u. a., 2006, S.18–22]. In den Klassen R1, R2, RC1 und RC2 sind die mit dem Verfahren ermittelten Ergebnisse gegenüber dem bekannten besten Optimum schlecht vergleichbar, da die Ergebnisse des genetischen Algorithmus meist mehr Fahrzeuge bei etwas geringeren Routenlängen enthalten.

THANGIAH teilt, ähnlich wie beim Sweep–Algorithmus, den Graph eines VRPTW ausgehend vom Depot in k Sektoren ein, wobei k die Anzahl der Fahrzeuge darstellt. Die zur Repräsentation einer Lösung verwendeten Chromosomen enthalten die Grenzen der k Sektoren in "Pseudo Polar–Koordinaten Winkeln", die Route eines Fahrzeugs wird jeweils durch einen Sektor repräsentiert [vgl. Tangiah, 1993, S.10–11]. Jede Sektorgrenze wird durch eine feste Anzahl von Bits im Chromosom kodiert. Dadurch ist prinzipiell die Anwendung der generischen Crossover–Operatoren (One–Point–Crossover, Two–Point–Crossover und Uniform–Crossover) möglich, dies wird von Thangiah jedoch nicht explizit geäußert. Der verwendete Crossover–Operator wird nicht explizit angegeben. Die Ecken innerhalb eines Sektors werden mit der Cheapest–Insertion Methode geroutet. Dabei wird aus den noch nicht gerouteten Ecken jeweils die Ecke ausgewählt, die mit den geringsten Kosten in die Route integriert werden kann, und an die entsprechende Stelle eingefügt. Nachdem die Routen innerhalb der Sektoren feststehen, wird die bestehende Lösung durch eine feste Anzahl von k–Opt Operationen weiter optimiert. Für diese Lösung werden die Sektoren neu berechnet und in den Chromosomen der Folgegeneration kodiert. Aus dem Artikel von Thangiah geht jedoch nicht hervor, wie mit Routen verfahren wird, die sich über mehrere Sektoren erstrecken [vgl. Tangiah, 1993]. Bei ungültigen Lösungen wird auf den Wert der Bewertungsfunktion ein "Strafwert" aufgeschlagen. Thangiah vergleicht sein Verfahren mit der Solomon Heuristik und der Heuristik von Thompson [Solomon:1986 und Thompson:1988, zit. n. Tangiah, 1993, S.17]. Die zitierte Dissertation von Thompson konnte nicht beschafft werden, eine Recherche zur Heuristik von Thompson blieb erfolglos. Die Heuristik von Solomon ist bei BENYAHIA UND POTVIN beschrieben [vgl. Benyahia und Potvin, 1995, S.2–3] und zählt zu den Konstruktionsheuristiken. Ein Vergleich mit anderen genetischen Verfahren oder Metaheuristiken wird von Thangiah nicht angegeben.

2.3.5.5 Large Neighbourhood Search

Unter den hier vorgestellten Metaheuristiken nimmt das 1997 von SHAW entwickelte Large Neighbourhood Search eine Sonderstellung ein, da zum einen bei der Suche im Lösungsraum bei insgesamt vergleichsweise wenigen Iterationen die aktuelle Lösung sehr stark modifiziert wird, zum anderen das Verfahren eine Hybridisierung aus Heuristiken und exakten Verfahren darstellt. Der Name Large Neighbourhood Search wird erst in Shaw [1998] verwendet.

Shaw führt zunächst das Problem der lokalen Minima bei der Suche mit lokalen Suchmethoden an, dem meistens durch Verwendung von Minima–Escaping Methoden begegnet wird [vgl. Shaw, 1997, S.1]. Enthält das zu optimierende Problem viele Randbedingungen, sind die durch einfache Operatoren modifizierten Lösungen oft ungültig.

In einem Vergleich zwischen der Methodik angewandter Algorithmen und der Methodik menschlicher Planer bei der Lösung von VRP stellt Shaw zwei hauptsächliche Unterschiede heraus: zum einen führen menschliche Planer im Gegensatz zu lokalen Suchmethoden an einer bestehenden Lösung kaum Veränderungen durch, die keine Verbesserung oder gar eine Verschlechterung der Lösung bewirken. Shaw führt dabei die eventuelle Anwendung von Voraussicht durch menschliche Planer an [3] [vgl. Shaw, 1997, S.1]. Zum anderen wählen menschliche Planer den Modifikationsoperator situationsbedingt aus ("... a human optimizer will tend to make up the neighbourhood as they go along" [Shaw, 1997, S.2]). In einigen Fällen können dies sehr einfache Operatoren sein. In anderen Fällen werden aufgrund der Randbedingungen zusammengesetzte bzw. komplexe Operatoren eingesetzt. Diese Operatoren erlauben es, grössere Bereiche des Suchraums von jedem Punkt aus zu erreichen und reduzieren somit die Notwendigkeit von Minima–Escaping Methoden [vgl. Shaw, 1997, S.2]).

Die von Shaw beschriebene Methode ist eine Greedy Suche, bei der nur bessere Lösungen als die aktuelle Lösung akzeptiert werden. Dabei wird eine Anzahl von Ecken aus den bestehenden Routen entfernt. Shaw verwendet eine gewichtete probabilistische Auswahl der zu entfernenden Ecken [vgl. Shaw, 1997, S.2]. Die entfernten Ecken werden durch ein Branch–and–Bound–Verfahren wieder in die bestehenden Routen eingesetzt. Die untere Schranke des Branch–and–Bound Verfahrens ist dabei der Wert, den die Bewertungsfunktion für die aktuelle Lösung liefert.

Zur Gewichtung wird eine *Relatedness Function* $\mathcal{R}(i,j)$ verwendet, die die Ähnlichkeit zweier Ecken bezüglich ihrer Entfernung voneinander sowie der Standzeit (bzw. Servicezeit) und Kapazitat des jeweiligen Requests definiert:

$$\mathcal{R}(i,j) = \frac{1}{\alpha c_{ij} + \beta|t_i - t_j| + \gamma T_{ij} + \delta|q_i - q_j|} \tag{2.13}$$

wobei c_{ij} die Kosten der Kante von i nach j, t_i und t_j der Start der Standzeit bei Ecken i und j sowie q_i und q_j die für die Ecken i und j benötigte Kapazität darstellen. T_{ij} beträgt 1, wenn beide Ecken in einer Route enthalten sind, andernfalls 0. Dabei sind c_{ij}, q_i, q_j, t_i, t_j auf den Bereich $[0,1]$ normiert. Durch die Parameter α, β und γ kann der Einfluss der einzelnen Aspekte gewichtet werden. Shaw gibt für seine Implementation die die Werte $\alpha = 0.75$,

[3] Einige lokale Suchmethoden versuchen diese Eigenschaft menschlicher Planer durch sogenannte *Regret–Heuristiken* abzubilden. Die im nächsten Kapitel vorgestellte Metaheuristik ALNS stellt ein Beispiel für die Anwendung von Regret–Heuristiken dar.

$\beta = \delta = 0.1$ und $\gamma = 1$ an [vgl. Shaw, 1997, S.3]. Zur Kontrolle der Diversifikation dient der Parameter p. Bei $p = 1$ ist die Auswahl ungewichtet probabilistisch. Für $p \to \infty$ werden die Ecken ausgewählt, bei denen der Wert der Relatedness Function maximiert wird. Shaw empfiehlt Werte von $3 \leq p \leq 5$ und verwendet in seiner Implementation den Wert $p = 4$.

Das folgende Schema gibt den Ablauf zur Auswahl der zu entfernenden Ecken wieder. Gegeben sei ein Graph $G(V, E)$ und eine Anzahl a zu entfernender Ecken.

1. $R \subset G$ und $R = \emptyset$.
2. Wähle zufallsgesteuert eine Ecke v_m aus. $R = R \bigcup v_m$, $G = G \backslash v_m$.
3. Wähle zufallsgesteuert eine Ecke v_n aus R.
4. Ordne alle nicht entfernten Ecken v_i des Graphen G bezüglich des Werts $\mathcal{R}(v_i, v_n)$ in einer Liste L an, ($L = G \backslash R$).
5. Wähle eine Zufallszahl r aus dem Intervall $[0, 1]$.
6. Wähle die Ecke v_o, die sich in der Liste L an der Stelle Integer(r^p) befindet. $R = R \bigcup v_o$.
7. Wenn $|R| < a$, weiter mit Schritt 3, ansonsten terminiere.

Das verwendete Branch–and–Bound Verfahren basiert auf Constraint Programming (CP). Bei jedem Schritt der Suche wird eine Variable instantiiert und die Contraint Propagierung durchgeführt. Durch die Propagierung ergibt sich meist eine Verkleinerung von Domänen einer oder mehrerer Variablen und somit auch eine Verkleinerung des Lösungsraums. Können eine oder mehrere Variablen nicht mit gültigen Werten belegt werden, wird Backtracking angewendet. Shaw gibt an, dass das Branch–and–Bound Verfahren bei ca. 25 Ecken in den meisten Fällen innerhalb weniger Sekunden eine bessere Lösung als die aktuelle Lösung zurück gibt. In einigen Fällen kann die Laufzeit jedoch um einiges höher ausfallen. Deshalb ist die Laufzeit des Branch–and–Bound Verfahrens auf 30 Sekunden begrenzt. Für Instanzen mit einer großen Anzahl von Ecken kann es nötig sein, eine unvollständige Suche durch heuristisches Beschneiden des Suchbaums vorzunehmen. Zu Anfang jeder Iteration der Optimierung werden 25 Ecken aus dem Graphen entfernt. Wird das Branch–and–Bound Verfahren bei Erreichen des Zeitlimits abgebrochen, ohne dass eine bessere Lösung gefunden wird, wird die Anzahl zu entfernender Ecken um eins verringert. Terminiert das Branch–and–Bound Verfahren bei 20 aufeinanderfolgenden Iterationen innerhalb des Zeitlimits, wird die Anzahl der zu entfernenden Ecken um eins erhöht.

Zur Einschätzung der Leistungsfähigkeit wendet Shaw seine Methode auf VRPTW Instanzen des Solomon Benchmarks (Klassen R1, C1 und RC1) sowie auf Instanzen des Benchmarks von Rochard und Taillard an. Für sechs Instanzen aus der Klasse R1 und zwei Instanzen aus der Klasse RC1 (jeweils Solomon Benchmark) werden dabei die zum Zeitpunkt des Tests

besten bekannten Lösungen verbessert [vgl. Shaw, 1997, S.8, 11–12]. Für drei Instanzen des Benchmarks von Rochard und Taillard werden die zum Zeitpunkt des Tests bekannten besten Lösungen verbessert [vgl. Shaw, 1998, S.9].

SHAW führt die Möglichkeit der Hybridisierung seiner Methode mit Minima–Escaping Verfahren an. Eine Hybridisierung mit Simulated Annealing oder Tabu–Suche wird zwar als möglich erachtet, aber in Frage gestellt, da durch das verwendete Branch–and–Bound Verfahren keine schlechteren Lösungen akzeptiert werden. Eine Hybridisierung mit weniger myoptischen Verfahren wie Guided Local Search wird hier als sinnvoller betrachtet [vgl. Shaw, 1997, S.6].

BENT UND VAN HENTENRYCK wenden eine Kombination aus Simulated Annealing und LNS als Zwei–Phasen–Verfahren auf das PDPTW an. In der ersten Phase wird durch Simulated Annealing die Anzahl der Routen minimiert, wobei als Modifikationsoperator ein einfacher Node–Pair Relocation Operator zum Einsatz kommt [vgl. Bent und Hentenryck, 2006, S.5]. In der zweiten Phase werden die bestehenden Routen hinsichtlich der Summe der Kantengewichte minimiert. Das von Bent und Hentenryck verwendete Verfahren zur Auswahl der Ecken ist eine auf das PDPTW angepasste Version des Auswahlverfahrens von Shaw [vgl. Bent und Hentenryck, 2006, S.8,9]. Bent und van Hentenryck wenden das Verfahren auf Instanzen des Benchmarks von Li und Lim an. Bei den Instanzen mit 100 Ecken wird in zwei Fällen die bisher beste bekannte Lösung verbessert, bei 54 weiteren Instanzen (93%) wird das bisher bekannte Optimum erreicht. Bei den Instanzen mit 200 Ecken wird in 28 Fällen (47%) die bisher beste bekannte Lösung verbessert, bei weiteren 24 Instanzen (40%) wird das bisher bekannte Optimum erreicht. Bei den Instanzen mit 600 Ecken wird in 46 Fällen (77%) die bisher beste bekannte Lösung verbessert, bei weiteren 5 Instanzen (8%) wird das bisher bekannte Optimum erreicht [vgl. Bent und Hentenryck, 2006, S.11-13].

2.3.5.6 Adaptive Large Neighbourhood Search

Das von ROPKE UND PISINGER in Ropke und Pisinger [2006a] vorgestellte Adaptive Large Neighbourhood Search (ALNS) stellt eine auf das PDPTW angepasste Weiterentwicklung des von Shaw entwickelten LNS dar. ALNS weicht in drei Aspekten vom LNS ab. Während bei Shaw bzw. Bent und van Hentenryck jeweils nur eine Heuristik bzw. ein begrenztes Branch–and–Bound Verfahren zur Entfernung bzw. zum Wiedereinfügen verwendet wird, werden beim ALNS sowohl zur Entfernung als auch zum Wiedereinfügen von Requests jeweils mehrere Heuristiken verwendet. Die Heuristiken zum Wiedereinfügen von Requests sind gegenüber dem Branch–and–Bound Algorithmus zum Wiedereinfügen von Ecken bei Shaw sowie Bent und van Hentenryck vergleichsweise einfach. Anstatt der deterministischen Akzeptanz von Lösungen wie beim LNS werden beim ALNS durch die Integration von Si-

mulated Annealing probabilistisch auch schlechtere Lösungen akzeptiert. [vgl. Ropke und Pisinger, 2006a, S.6].

ROPKE UND PISINGER bezeichnen ALNS als ein Framework, da sowohl die Heuristik auf dem "Master Level" [Pisinger und Ropke, 2007, S.14] als auch die Heuristiken zum Entfernen und Wiedereinfügen von Requests ausgetauscht werden können. Durch die verschiedenen Heuristiken, die jeweils zum Entfernen bzw. Wiedereinfügen von Requests verwendet werden, werden verschiedene Nachbarschaften untersucht. Ropke und Pisinger nennen die Nachbarschaften *Destroy Neighbourhood* (Entfernung von Requests) und *Repair Neighbourhood* (Wiedereinfügen von Requests). Für jede Entfernungs– und Einfüge–Heuristik h_i wird die Perfomanz innnerhalb einer bestimmten Anzahl der letzten Iterationen festgehalten, der Wert dafür wird mit π_i bezeichnet. Die Heuristiken werden in jeder Iteration gemäß ihrer Perfomance π_i durch die aus dem Bereich der genetischen Algorithmen stammenden *Roulette Wheel Selection* ausgewählt. Das grobe Ablaufschema des ALNS sieht wie folgt aus:

1. Konstruiere eine gültige Startlösung x. Setze die bisher beste gefundene Lösung $x^* = x$.
2. Wähle anhand der Performanz π_i und π_j mit der Roulette Wheel Selection eine Entfernungsheuristik h_i und eine Einfügeheuristik h_j.
3. Generiere durch Modifiktion der Lösung x durch Anwendung von h_i und h_j eine Lösung x^t.
4. Wird x^t durch die Heuristik auf dem Master Level akzeptiert, setze $x = x^t$.
5. Wenn $f(x) < f(x^*)$, dann setze $x^* = x$.
6. Wenn die Abbruchbedingung erfüllt ist, gib x^* zurück und terminiere. Ansonsten weiter mit Schritt 2.

Im folgenden wird auf die Entfernungsheuristiken eingegangen. In Ropke und Pisinger [2006a] werden drei Entfernungsheuristiken vorgestellt, vier weitere Entfernungsheuristiken werden in Pisinger und Ropke [2007] vorgestellt. Von den letzteren werden hier nur zwei detailliert vorgestellt, die anderen beiden Heuristiken sind Abwandlungen der hier beschriebenen Heuristiken. Allen Heuristiken wird als Eingabe eine Lösung s, die Anzahl $q \in \mathbb{N}$ zu entfernender Requests sowie ein Paramter $p \in \mathbb{R}_+$, durch den der Grad der Randomisierung der Heuristik gesteuert wird, übergeben.

Bei der von Shaw beschriebenen Heuristik zur Entfernung von zwei ähnlichen Requests (siehe Formel 2.13) wird von Ropke und Pisinger die Relatedness Function $\mathcal{R}(i,j)$ auf das PDPTW passend abgewandelt:

$$\mathcal{R}(i,j) = \phi(d_{A(i),A(j)} + d_{B(i),B(j)}) + \chi(|T_{A(i)} - T_{A(j)}| + |T_{B(i)} - T_{B(j)}|) + \Psi|l_i - l_j| + \omega\left(1 - \frac{|K_i \bigcap k_j|}{\min\{|K_i|, |K_j|\}}\right) \tag{2.14}$$

wobei $A(i)$ und $B(i)$ die Pickup–Punkte, $A(j)$ und $B(j)$ die Delivery–Punkte der Requests i und j darstellen. T_m ist der Zeitpunkt, zum dem der Punkt m angefahren wird. l_i ist die Kapazität des Requests i, K_i ist die Menge der Fahrzeuge, die zur Abwicklung des Requests i benutzt werden können. Die Parameter ϕ, χ, Ψ und ω dienen zur Gewichtung der einzelnen Anteile der Relatedness Function. Der erste Term enthält die Entfernung der beiden Pickup–Punkte und der beiden Delivery–Punkte. Der zweite Term enhält die zeitliche Entfernung der beiden Pickup–Punkte und Delivery–Punkte, der dritte Term enthält den Unterschied der Kapazität der beiden Requests. Durch den vierten Term wird erreicht, dass nur diejenigen Reuests eine hohe Relatedness erhalten, wenn nur wenige Fahrzeuge oder kein Fahrzeug beide Requests bedienen können. d_{ij}, T_m und l_i werden auf den Bereich $[0, 1]$ normalisiert. Der Algorithmus zur Entfernung von Requests ist dabei der gleiche wie bei Shaw, die Heuristik wird von Ropke und Pisinger als *Shaw Removal* bezeichnet.

Beim *Random Removal* wird eine Anzahl von Requests zufallsgesteuert entfernt. Dasselbe Resultat könnte auch mit der Shaw Removal Heuristik bei $p = 1$ erreicht werden, allerdings ist der Algorithmus des Random Removal um einiges einfacher und die entsprechende Implementation ist schneller als die des Shaw Removal.

Durch den *Worst Removal* werden iterativ diejenigen q Requests entfernt, die in der gegenwärtigen Lösung am meisten Kosten verursachen. Dabei werden vor jeder Entfernung eines Requests die Kosten aller Requests neu berechnet und in einem Array L in absteigender Reihenfolge bezüglich der Kosten sortiert abgelegt. Die Auswahl des jeweils zu entfernenden Requests erfolgt probabilistisch, der Index r des zu entfernenden Requests ist durch $r = y^p|L|$ gegeben, wobei y eine Zufallszahl aus dem Intervall $[0, 1]$ ist.

Beim *Historical Request–Pair Removal* werden jeweils paarweise die Requests entfernt, die sich in den letzten B Iterationen am häufigsten in der Route des gleichen Fahrzeugs befanden. Dazu wird für die Requests eine Gewichtsmatrix eingeführt. Das Gewicht eines Requests $r_{A,B}$ ist durch die Anzahl an Lösungen der letzten B Lösungen bestimmt, innerhalb deren die Requests A und B sich gemeinsam in der Route eines Fahrzeugs befanden. Ein günstiger Wert für B wurde von Ropke und Pisinge empirisch ermittelt und mit $B = 100$ angegeben [vgl. Pisinger und Ropke, 2007, S.13].

ROPKE UND PISINGER führen an, dass das Entfernen von einigen Requests aus einem Cluster von Requests meist unvorteilhaft ist, da zur Generierung einer neuen, qualitativ guten Lö-

sung die entfernten Request meist nur wieder in diejenige Route eingefügt werden können, aus der sie entfernt wurden. In diesen Fällen ist es vorteilhafter, ganze Cluster von Requests zu entfernen. Die entsprechende Heuristik für die Anwendung auf VRP wird in [Ropke und Pisinger, 2006b, S.11] beschrieben, die Anpassung auf PDPTW wird in [Pisinger und Ropke, 2007, S.12] kurz umrissen. Beim *Cluster Removal* wird eine Route zufallsgesteuert ausgewählt. Die in der Route enthaltenen Requests werden bezüglich ihrer jeweils paarweisen Entfernung zueinander in zwei Cluster aufgeteilt. Dazu wird der Algorithmus von Kruskal zur Bestimmung von Minimalgerüsten [4] auf die Requests der Route angewendet und (vorzeitig) terminiert, wenn alle Requests der Route in zwei zusammenhängenden Komponenten zusammengefasst sind. Werden durch diese Operation weniger als q Requests entfernt, wird zufallsgesteuert ein Request r_i aus einer der beiden entstandenen Komponenten ausgewählt und derjenige Request r_j entfernt, der zu r_i am ähnlichsten ist. Abbildung 2.13 verdeutlicht den Vorgang beim Cluster Removal.

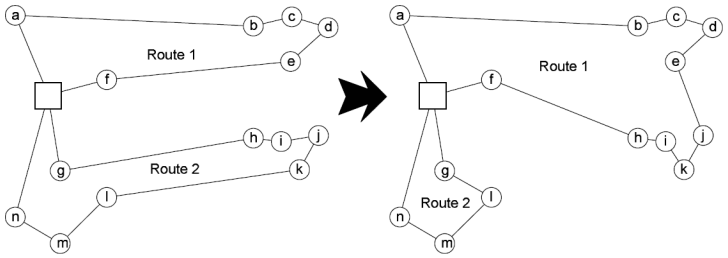

Abbildung 2.13: Cluster Removal: originale Lösung auf der linken und resultierende Lösung auf der rechten Seite. Graphik aus Ropke und Pisinger [2006b]

Für das Einfügen von Requests in bestehende Lösungen stellen ROPKE UND PISINGER zwei Heuristiken vor. Bei der *Basic Greedy Heuristik* werden nacheinander aus den entfernten Requests diejeingen ausgewählt, bei denen das Einfügen an die jeweils für den Request günstigste Position die Kosten der bestehenden Lösung am wenigsten erhöht [Pisinger und Ropke, 2007, S.13]. Die Requests werden an der entsprechenden Position eingefügt. Allerdings tritt dabei die Problematik auf, dass das Einfügen von Requests mit höheren Kosten auf einen späteren Zeitpunkt verlagert wird, zu dem die entsprechenden Request nur noch an sehr ungünstigen Positionen eingefügt werden können. Diese Problematik lässt sich auch gut anhand des Nearest Neighbour Algorithmus zur Generierung von TSP–Touren verdeutlichen. Bei der *Regret Heuristik* wird versucht, diese Problematik zu vermeiden. Sei $x_{ra} \in 1, \ldots, m$ die Variable, die die a–beste Route k bezeichnet, in die der Request r eingefügt werden kann (x_{r1} bezeichnet die Route, in die der Request r am kostengünstigsten eingefügt werden kann, x_{r2} bezeichnet die zweitbeste Route bezüglich der Einfügekosten, etc.), wobei m die Anzahl der Routen ist. Als Einfügekosten wird die Kostenerhöhung Δf_r^a

[4] zur Beschreibung des Kruskal Algorithmus zur Bestimmung von Minimalgerüsten siehe u. a. [Klauck und Maas, 1999, S.82,83].

bezeichnet, um die die Kosten der gegenwärtigen Lösung durch Einfügen des Requests r an die günstigste Position in der a–besten Route x_{ra} erhöht werden. Bei der *Regret-a Heuristik* werden beim Einfügen nacheinander die Requests ausgewählt, durch die die Summe der Einfügekosten Δf_r^a mit $a \in 1, \ldots, m$ maximiert wird:

$$r = \arg\max_{i \in U} \left(\sum_{a=2}^{m} \Delta f_r^a - \Delta f_r^1 \right) \tag{2.15}$$

wobei U die Menge der Requests ist, die in der gegenwärtigen Lösung nicht enthalten sind. Die ausgewählten Requests werden jeweils an die günstigste Position innerhalb der gegenwärtigen Lösung eingefügt.

Bei der Suche im Lösungsraum werden in jeder Iteration je eine Entfernungs– und eine Einfüge–Heuristik probabilistisch, nach Leistungsfähigkeit der Heuristiken gewichtet, ausgewählt. Bei n Heuristiken mit den Gewichten $w_i, i \in 1, \ldots, n$, wird eine Heuristik j mit der Wahrscheinlichkeit

$$P(j) = \frac{w_j}{\sum_{i=1}^{n} w_i} \tag{2.16}$$

ausgewählt. Die Suche im Lösungsraum ist in *Segmente* aufgeteilt, wobei ein Segment eine Anzahl von Iterationen umfasst. ROPKE UND PISINGER geben die Länge eines Segments mit 100 Iterationen an [vgl. Ropke und Pisinger, 2006a, S.9]. Das Gewicht einer Heuristik wird durch eine Punktezahl repräsentiert, die am Anfang des ersten Segments für alle Heuristiken auf Null gesetzt wird. Die Punktezahl ausgewählten Entfernungs– bzw. Einfüge–Heuristik wird bei einer der folgenden Situationen um einen der Werte σ_1, σ_2 bzw. σ_3 erhöht:

- σ_1: die letzte Entfernungs– und Einfügeoperation ergab eine neue global beste Lösung.
- σ_2: die letzte Entfernungs– und Einfügeoperation ergab eine Lösung, die bisher noch nicht akzeptiert wurde. Die Kosten der neuen Lösung sind niedriger als die Kosten als der vorherigen Lösung.
- σ_3: die letzte Entfernungs– und Einfügeoperation ergab eine Lösung, die bisher noch nicht akzeptiert wurde. Die Kosten der neuen Lösung sind höher als die Kosten der vorherigen Lösung. Die neue Lösung wurde allerdings akzeptiert.

Am Anfang eines jeden Segments $j + 1$ werden die Gewichte aller Heuristiken neu berechnet:

$$w_{i,j+1} = w_{ij}(1 - r) + r\frac{\pi_i}{\theta_i} \tag{2.17}$$

wobei π_i die von der Heuristik h_i im Segment j erreichte Punktezahl ist und θ_i anzeigt, wie oft die Heuristik h_i im Segment j angewendet wurde. Durch den Faktor r wird bestimmt, wie schnell sich Änderungen in der Performanz von Heuristiken auf deren Punktezahl auswirken, wobei $0 \leq r \leq 1$ [vgl. Ropke und Pisinger, 2006a, S.10].

Ein $günstiger$ Wertebereich für die Anzahl q an Requests, die bei einer Modifikation der aktuellen Lösung entfernt und wieder eingefügt werden, wird von ROPKE UND PISINGER empirisch ermittelt und mit $[\min 0.1n, 30, \min 0.4n, 60]$ angegeben, wobei n die Anzahl an Requests darstellt. Bei jeder Iteration wird die Anzahl der zu entfernenden Requests durch eine zufallsgenerierte Zahl r aus dem Intervall $[\min 0.1n, 30, \min 0.4n, 60]$ bestimmt [vgl. Pisinger und Ropke, 2007, S.16].

Der Bewertungsfunktion liegt ein Modell zugrunde, bei dem für jedes Fahrzeug jeweils ein beliebiges Start– und Enddepot aus einer Menge von Depots vergeben werden kann. Ausserdem ist die Bewertung von Lösungen, in denen nicht alle Requests enthalten sind, möglich. In der aktuellen Lösung nicht enthaltene Requests werden in einer *Request Bank* gespeichert. Die Kosten einer Lösung werden durch folgende Bewertungsfunktion f bestimmt:

$$f = \alpha \sum_{k \in K} \sum_{(i,j \in A)} d_{ij} x_{ijk} + \beta \sum_{k \in K} \left(S_{\tau_k^*,k} - a_{\tau_k} \right) + \gamma \sum i \in P z_i \tag{2.18}$$

wobei

- K die Menge aller Fahrzeuge,
- A die Menge aller Kanten ($V \times V$),
- d_{ij} das Gewicht der Kante mit Anfangsecke i und Endecke j,
- die binäre Entscheidungsvariable $x_{ijk} = 1$, wenn das Fahrzeug k ($k \in K$), Kante x_{ij} benutzt, ansonsten $x_{ijk} = 0$,
- $S_{\tau_k^*,k}$ der Zeitpunkt, an dem das Fahrzeug k das ihm zugehörige Enddepot τ_k^* befährt,
- a_{τ_k} der früheste Zeitpunkt, zu dem das Fahrzeug k das Startdepot τ_k verlassen kann,
- P die Menge der Pickup–Punkte,
- die binäre Entscheidungsvariable $z_i = 1$, wenn der Pickup–Punkt $i \in P$ in keiner Route der Lösung enthalten ist (der entsprechende Request befindet sich in der *Request Bank*), ansonsten $z_i = 0$
- und α, β sowie γ die Parameter zu Gewichtung der einzelnen Teile der Bewertungsfunktion darstellen.

Der erste Teil der Bewertungsfunktion enthält die Summe der Kantengewichte bezüglich der Streckenlänge, der zweite Teil enthält die Summe der von allen Fahrzeugen für ihre Routen benötigten Zeit und der dritte Teil enthält die nicht berücksichtigten Requests [vgl. Ropke und Pisinger, 2006a, S.3].

Zur Diversifikation wird das Ergebnis der Bewertungsfunktion "verrauscht". Bei jeder Berechnung der Kosten C beim Einfügen eines Requests in eine Route wird ein Zufallswert $noise$ aus dem Interval $[-\max N, \max N]$ bestimmt und die modifizierten Kosten $C^* = \max 0, C + noise$ berechnet. Bei jeder Iteration wird entschieden, ob die realen Kosten C oder die verrauschten Kosten C^* zur Entscheidung über die Akzeptanz verwendet werden sollen. Die Entscheidung wird dabei durch das gleiche adaptiven Verfahren bestimmt wie das Verfahren zur Auswahl der Entfernungs– und Einfüge–Heuristiken. Damit die Stärke des Rauschens mit Eigenschaften der konkreten Probleminstanz korreliert, wird $\max N = \eta \cdot \max_{i,j \in V} d_{ij}$ gesetzt, wobei der Parameter η die Menge des Rauschens bezeichnet. Ropke und Pisinger geben den Wert $\eta = 0.025$ an [vgl. Ropke und Pisinger, 2006a, S.11,13].

ALNS wird in zwei Phasen auf Instanzen eines PDPTW angewendet. In der ersten Phase wird die Anzahl der Fahrzeuge minimiert, in der zweiten Phase werden die Längen der Routen minimiert. Ropke und Pisinger führen dabei an, dass die Minimierung der Anzahl an Fahrzeugen nur bei homogenen Fahrzeugflotten möglich ist. Zur Minimierung der Anzahl an Fahrzeugen startet die erste Phase mit einer durch eine Konstruktionsheuristik generierten Startlösung. Von dieser Startlösung wird eine Route entfernt und die Requests dieser Route in der Request Bank gespeichert. ALNS startet mit der verbleibenden Startlösung und den Requests in der Request Bank. Dieser Prozess wird iterativ solange wiederholt, bis innerhalb einer bestimmten Anzahl von Iterationen keine gültige Lösung gefunden werden kann. In diesem Fall wird zu der vorherigen Lösung zurückgekehrt und die zweite Phase zur Minimierung der Routen bezüglich Zeitdauer und Streckenlänge gestartet.

Die Leistungsfähigkeit von ALNS wurde mit Hilfe des Benchmarks von Li und Lim evaluiert. In der Klasse mit 200 Requests pro Instanz konnte von den 50 Instanzen der Klasse bei 27 Instanzen die bisher bekannte beste Lösung verbessert werden. In der Klasse mit 400 Requests konnten von den 60 Instanzen der Klasse bei 41 Instanzen die bisher bekannte beste Lösung verbessert werden. In den Klassen mit 600, 800 und 1000 Requests konnten bei jeweils ca. $\frac{5}{6}$ der Instanzen die bisher bekannte beste Lösung verbessert werden. Insgesamt wurde bei allen Instanzen des Benchmarks bis auf wenige Ausnahmen die bisher bekannte beste Lösung entweder erreicht oder verbessert.

2.3.5.7 Agentenbasierte Systeme

Die Literaturrecherche zu agentbasierten Systemen ergab nur wenige relevante Artikel, insgesamt ist der Bereich der agentenbasierten Systeme im VRP–Bereich eher überschaubar.

KOHOUT UND EROL stellen ein System zum Online–Routing einer Flotte von Flughafen–Shuttles vor [vgl. Kohout und Erol, 1999]. Bei dem Multiagentensystem wird jedes Fahrzeug und jeder Kunde durch je einen Agenten repräsentiert. Die Verhandlungen der Agenten basieren stark auf einer auf das PDPTW angepassten Version eines klassischen Konstruktionsalgorithmus, des Solomon Algorithmus. Anhand eines Auszugs der Datenbasis eines Flughafen–Shuttle–Betriebs in der Nähe von Washington, DC wurden stochastisch zwei Sets von Testdaten generiert. Ein Set von 100 Instanzen enthält je 100 Kunden, ein Set von ebenfalls 100 Instanzen enthält je 200 Kunden. Das agentenbasierte System ist in zwei Versionen implementiert, eine Version enthält im Gegensatz zur anderen Version eine zusätzliche stochastische Optimierung. In Testläufen wurde die Performanz beider Versionen gegenüber der auf das PDPTW angepassten Version des Solomon Algorithmus ermittelt. Die agentenbasierte Version ohne zusätzliche stochastische Optimierung schneidet dabei sehr schlecht ab, die Anzahl an benötigten Fahrzeugen ist im Mittel 25,6% (Instanzen mit 100 Kunden) bzw. 31% (Instanzen mit 200 Kunden) höher als bei Lösungen des Solomon Algorithmus. Die totale Routendauer der Kunden liegt jeweils 35,6% bzw. 43,1% über den Ergebnissen des Solomon Algorithmus. Die agentenbasierte Version mit stochastischer Optimierung schneidet um einiges besser ab und ergibt im Mittel bezüglich der Anzahl an benötigten Fahrzeugen um ca. 2,4% bessere Ergebnisse als der Solomon Algorithmus. Allerdings sind die Routenkosten um 6,9% (Instanzen mit 100 Kunden) bzw. 2% (Instanzen mit 200 Kunden) höher als bei den Ergebnissen des Salomon Algorithmus [vgl. Kohout und Erol, 1999, S.5,6].

BOUDALI U. A. stellen ein agentbasiertes System namens COAL–VRP vor, bei dem die Agenten in Gruppen, die als *Koalitionen* (coalitions) bezeichnet werden, zusammengefasst sind [Boudali u. a., 2004, vgl.] sowie Boudali u. a. [2005]. Die Agenten agieren innerhalb der Koalition, der sie zugeordnet sind, können aber diese auch verlassen und sich einer neuen Koalition anschliessen. Eine Implementation des Models wird anhand einiger Instanzen des Solomon Benchmarks, R1, C1 und RC1 mit 50 Kunden getestet. Bei den Tests sind die Ergebnisse des COAL–VRP Systems durchweg mindestens 10% über dem bekannten Optimum. In den meisten Fällen liegen die Ergebnisse mehr als 20% – 30% über dem Optimum, in einigen wenigen Fällen mehr als 40% über dem Optimum [vgl. Boudali u. a., 2005, S.48/9, 48/10].

In den Übersichten über klassische und moderne Heuristiken von LAPORTE U. A. sowie über Metaheuristiken für das VRP von GENDREAU U. A. werden agentenbasierte Systeme nicht erwähnt [Laporte u. a., 2000, vgl.] und [Gendreau u. a., 2001, vgl.].

2.4 Lösungsansätze und Verfahren für dynamisch stochastische VRP

Bei der Behandlung von statischen VRP werden die Informationen berücksichtigt, die zum Zeitpunkt der *a priori* Planung vorliegen. Das dynamisch–stochastische VRP erweitert das statische VRP um eine weitere Dimension, die Dimension der Zeit. Die Qualität eines bestimmten Plans kann sich innerhalb des Planungshorizonts ändern. Beim dynamischen Flottenmanagement wird, zusätzlich zu einer *a priori* Planung, während des Planungshorizonts der bestehende Plan durch Entscheidungen aufgrund von Informationen, die über einen bestimmten Zeitraum vorliegen, geändert oder erweitert. Die Informationen beziehen sich dabei nicht nur auf die konkrete Ausprägung des Problems, wie sie zum Zeitpunkt der *a priori* Planung bekannt ist, sondern auch auf das Auftreten von Events. Informationen über vergangene Events können in Form eines Datenbestands oder als Erfahrungswissen eines menschlichen Planers (Dispatcher) vorliegen. Das zukünftige Auftreten von Events kann nur abgeschätzt werden. Bestenfalls liegen Informationen über die Wahrscheinlichkeit des zukünftigen Auftretens von Events als stochastische Verteilungen vor.

Während bei statischen VRP die Bewertung einer Lösung anhand der Kosten der Lösung erfolgen kann, ist bei der Bewertung einer Lösung für ein dynamisch–stochastisches VRP die Fähigkeit zur angemessenen Reaktion auf Events als weiteres Kriterien vorhanden.

ICHOUA U. A. bemerken, dass sich die Aktivitäten auf dem Gebiet des Echtzeit–Managements von Fahrzeugflotten in den letzten Jahren verstärkt haben. Die Autoren sehen als Ursachen die wirtschaftliche und technologische Entwicklung der letzten Jahre an. Einerseits führen die wirtschaftlichen Entwicklungen zu offeneren Märkten mit höherem Konkurrenzdruck, andererseit ermöglicht der Fortschritt in den Kommunikationstechnologien einen kostengünstigeren Zugang zu Echtzeit–Informationen [vgl. Ichoua u. a., 2007, S.].

Die Verfügbarkeit von Echtzeit–Informationen wie z. B. über die aktuelle Positionen von Fahrzeugen der Flotte und Änderung von Einflüssen wie Verkehrsaufkommen oder Wetter ermöglichen eine Abschätzung der Entwicklung des Szenarios während des Planungshorizonts. In Verbindung mit Informationen aus der Vergangenheit lassen sich auch Trends abschätzen. Der Einfluss bestehender Technologien auf das dynamische Flottenmanagement wird durch aktuelle Designs von Systemen für das dynamische Flottenmanagement deutlich. ZEIMPEKIS U. A. implementieren ein System für einen 3PL Operator (3rd Party Logistics Operator) in der Nähe von Athen, Griechenland, dessen Flotte täglich mehr als 150 Tonnen an Gütern von ca. 300 Kunden transportiert, wobei die Kunden zwischen drei (3) und vierzig (40) Kilometer vom Standord der Firma entfernt sind. Zur Handhabung kartographischer Informationen ist ein geographisches Informationsmodul (GIM) in das System intergriert. Die Positionsbestimmung der Fahrzeuge und die Kommunikation zwischen Fahrer und zentralem Depot erfolgt

durch ein in die Fahrzeuge integriertes und auf dem General Packet Radio System basierenden Terminal. Abbildung 2.14 zeigt die Architektur des Systems [vgl. Zeimpekis u. a., 2007, S.201, 208].

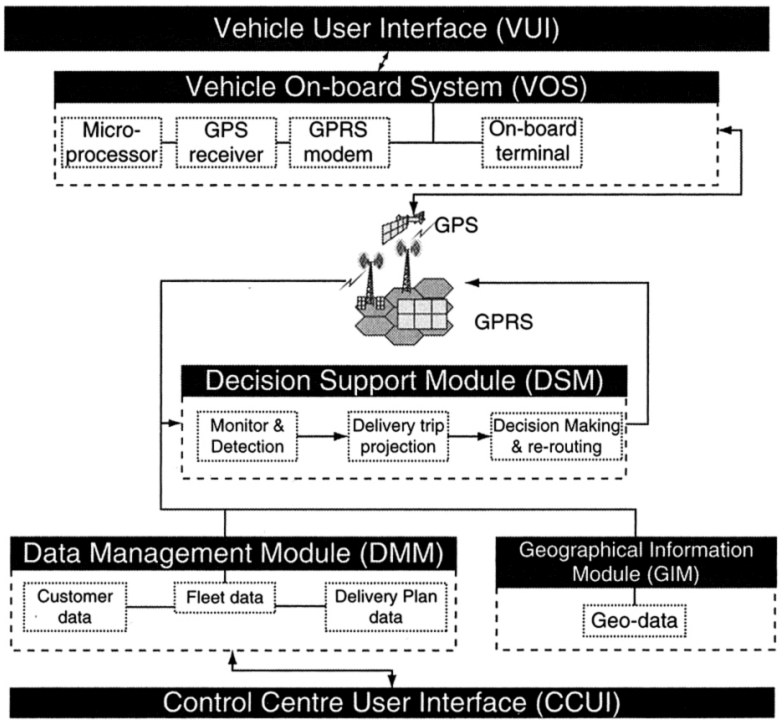

Abbildung 2.14: Architektur eines Systems zum dynamischen Flottenmanagement nach Zeimpekis. Graphik aus Zeimpekis u. a. [2007].

2.4.1 Quantitative Erfassung der Dynamik von DSVRP

Der Grad der Dynamik eines DSVRP ist je nach Umfeld variabel. Bei Transportunternehmen sind je nach Umfeld schon vor Beginn des Planungshorizonts Requests bekannt. Die Einsatzplanung von Fahrzeugen im administrativen Bereich zur Behandlung von Notrufen, wie z. B. bei Polizei, Feuerwehr und Ambulanz ist rein dynamisch, da Notrufe nicht im vorhinein bekannt sind. Zur Einschätzung der jeweiligen Problematik ist eine quantitative Erfassung der Dynamik eines Problems vorteilhaft. LARSEN U. A. nennen als erste Definition der Dynamik eines VRP den in der Arbeit von LUND U. A. definierten *Degree of Dynamism* (Grad der Dynamik). Dieser ist als das Verhältnis der Anzahl von spontanen Requests n_{imm} (*immediate Requests*) zur gesamten Anzahl von Requests n_{tot} (*total Requests*) definiert [Lund:1996, zit. n. Larsen u. a., 2007]:

$$dod = \frac{n_{imm}}{n_{tot}} \tag{2.19}$$

Die Berücksichtigung spontaner Requests wird jedoch umso schwieriger, je später diese innerhalb des Planungshorizonts auftreten. LARSEN U. A. definieren den *extended Degree of Dynamism* (edod) als:

$$edod = \frac{\sum_{i=1}^{n_{imm}} \frac{t_i}{T}}{n_{tot}} \tag{2.20}$$

wobei der Planungszeitraum als der Zeitraum zwischen den Zeitpunkten 0 und T definiert ist. t_i bezeichnet den Zeitpunkt, an dem der spontane Request i bekannt wird, mit $0 \leq i \leq T$. Daraus folgt, dass $0 \leq edod \leq 1$. Ein System mit einem edod von 0 wird als rein dynamisch und eine System mit einem edod von 1 wird als rein statisch angesehen [vgl. Larsen u. a., 2007, S.29]. Sind bei der Abwicklung von Requests Zeitfenster zu berücksichtigen, sind diejenigen Requests einfacher in einen bestehenden Plan zu integrieren, die relativ früh bekannt werden, deren Ende des Zeitfensters relativ weit von dem Zeitpunkt entfernt sind, zu dem der Request bekannt wird und deren Zeitfenster relativ groß sind. Für VRP mit Zeitfenstern definieren LARSEN U. A. den $edod_{tw}$ als:

$$edod_{tw} = \frac{1}{n_{tot}} \sum_{i=1}^{n_imm} \frac{T - (l_i - t_i)}{T} = \frac{1}{n_{tot}} \left(1 - \frac{r_i}{T}\right) \tag{2.21}$$

wobei l_i das Ende des Zeitfensters des Requests i darstellt. Wiederum gilt $0 \leq edod_{tw} \leq 1$.

2.4.2 Klassifizierung von DSVRP

LARSEN U. A. schlagen ein Framework zur Klassifizierung von DSVRP vor, bei dem die DSVRP anhand ihres jeweiligen edod sowie anhand des operationalen Ziels eingeordnet werden. Das operationale Ziel enthält zwei sich diametral gegenüberliegende Aspekte: die Minimierung der Antwortzeit (Zeitraum zwischen Bekanntwerden des Requests und der Ankunft des Servicefahrzeugs beim Kunden) und die Minimierung von Kosten zur Abwicklung eines Requests. [vgl. Larsen u. a., 2007, S.34–37]. Die DSVRP werden anhand ihres edod in drei Klassen eingeteilt:

- Echelon I – schwach dynamische Systeme: die meisten DSVRP, die die Verteilung von Waren beinhalten, fallen in diese Klasse. LARSEN U. A. nennen als Beispiel die Verteilung von Heizöl, bei der die meisten Requests vor dem Planungshorizont bekannt sind,

jeoch einige spontane Requests von Kunden auftreten können, denen das Heizöl ausgegangen ist. In diesen Fällen sollte die Belieferung möglichst schnell erfolgen.

- Echelon II – mittel dynamische Systeme: hierunter fallen Systeme mit einer relativ großen Anzahl von spontanen Requests. LARSEN U. A. nennen als Beispiel u. a. die Bestückung und Wartung von Geldautomaten.
- Echelon III – stark dynamische Systeme: in diese Klasse fallen Systeme, bei denen alle bzw. fast alle Requests während des Planungshorizonts auftreten. Als Beispiele sind die Einsatzplanung bei Notrufen oder das Routing von Taxis zu nennen.

Abbildung 2.15 zeigt das Framework zur Klassifizierung mit Beispielen von DSVRP.

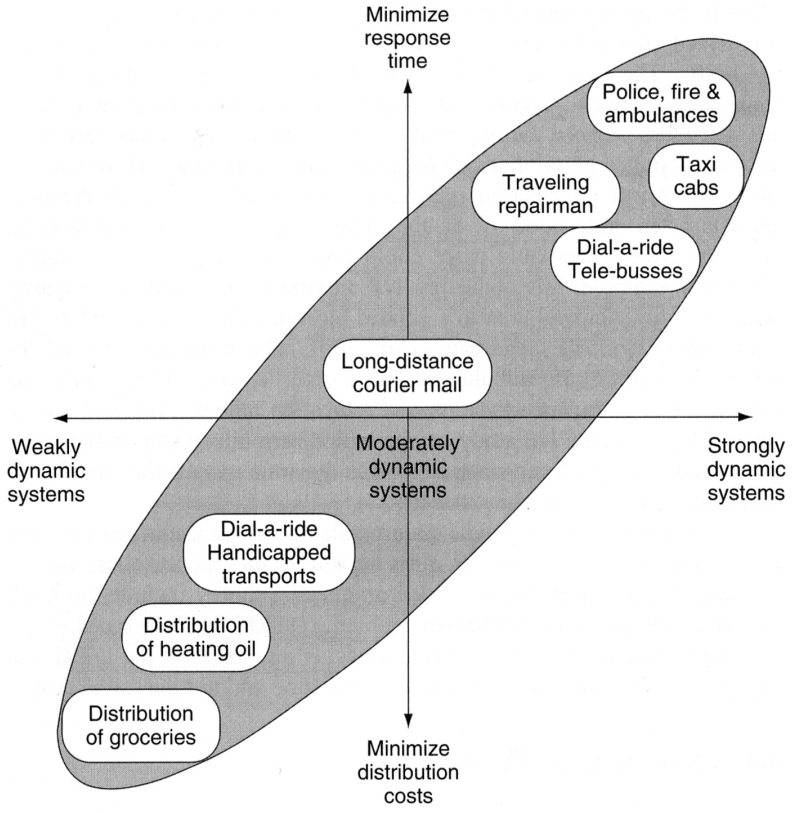

Abbildung 2.15: Framework zur Klassifizierung von DSVRP nach Grad der Dynamik. Graphik aus Larsen u. a. [2007].

2.4.3 Abschätzung der Entwicklung von Szenarios

Eine realistische Abschätzung der Entwicklung des Szenarios bezüglich des Auftretens von Events bzw. der Entwicklung anderer relevanter Einflüsse ist eine wichtige Grundlage für Routing–Entscheidungen, die während des Planungshorizonts getroffen werden. Durch Entscheidungen, die zukünftige Events und die Entwicklung relevanter Einflüsse antizipieren, werden Pläne gegenüber tatsächlich stattfindenden Events robuster.

Einen abstrakten Ansatz zur Verwendung von Prognosen und historischem Wissen über Events stellen FLATBERG U. A. vor. Das Verfahren zur Optimierung von DSVRP speichert sowohl den initialen Zustand des DSVRP als auch die während des Planungshorizonts auftretenden Events in sogenannten *Event cases* ab. Diese werden zur Generierung von probabilistischem Wissen über das zukünftige Auftreten von Events benutzt, durch welches wiederum das Verfahren zur Optimierung von DSVRP beeinflusst wird. Zusätzlich besteht die Möglichkeit der Anbindung von externen Quellen, die Prognosen zur Verfügung stellen. Abbildung 2.16 veranschaulicht den Zusammenhang. Der Ansatz ist in dem kommerziellen Produkt Spider DSVRP Solver implementiert, das von Flatberg u. a. auch zur wissenschaftlichen Forschung eingesetzt wird [vgl. Flatberg u. a., 2007, S.50, 53].

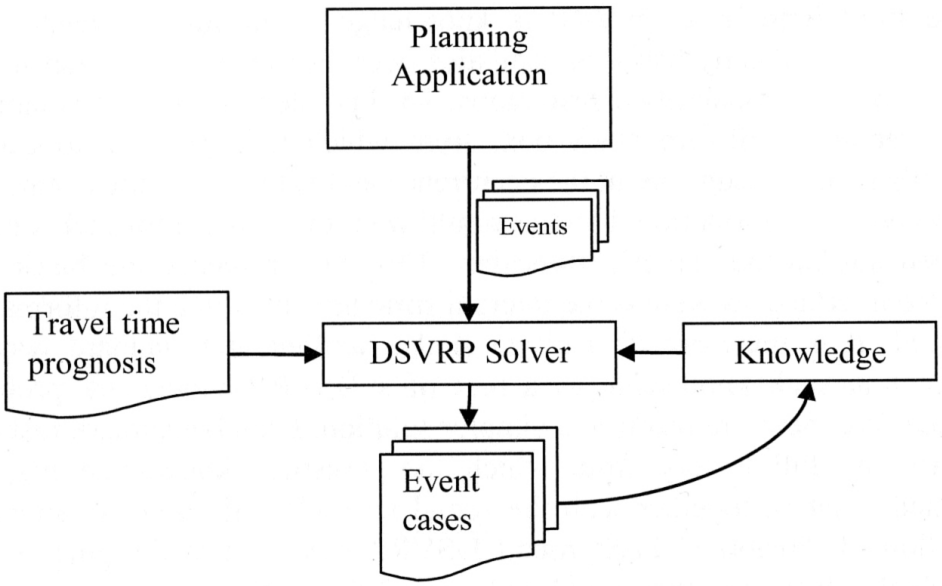

Abbildung 2.16: Kontext eines Verfahrens zum Lösen von DSVRP. Graphik aus Flatberg u. a. [2007].

Konkrete Verfahren zur Erstellung von Prognosen bezüglich künftiger Pickup– and Delivery–Requests und Zeitbedarf für Streckenabschnitte werden von ATTANASIO U. A. angegeben.

Das System zur Prognose ist in das System zum dynamischen Flottenmanagement der Firma eCourier LTD, London, UK integriert und basiert auf einer Modifikation der Methode der *klassischen Dekomposition* von MONTGOMERY U. A. [Montgomery:1990, zit. n. Flatberg u. a., 2007, S.224]. Die Prognose zukünftiger Requests wird aufgrund des hohen Aufkommens an Requests nicht als absolute Anzahl, sondern als Request Rate (Requests pro Minute) berechnet.

Bei der Prognose werden die saisonalen Effekte des betreffenden Zeitraums im Jahr, des Wochentages sowie der Tageszeit berücksichtigt. Dabei ist ein Jahr in unterschiedlich lange Zeiträume zerlegt, deren Grenzen sich nicht an Monatsgrenzen, sondern an saisonalen Abschnitten wie z. B. “New Year time” (nach Weihnachten bis Mitte Februar), “End of Financial Year time” (Mitte März bis 5. April), “Summer period” (Mitte Mai bis Mitte Juli) und “Holiay Season” (Ende Juli bis Ende August) orientieren. Die Grenzen der Zeiträume für Wochentage und Tageszeiten sind äquidistant [vgl. Attanasio u. a., 2007, S.222–223]. Zusätzlich zu den saisonalen Effekten wird “Trend” durch die Gaußsche Methode der kleinsten Quadrate berechnet. Die Prognose wird durch Multiplikation der saisonalen Effekte und des Trends berechnet.

Das Verfahren für die Berechnung des Zeitbedarfs für Streckenabschnitte ist ähnlich dem Verfahren zur Prognose von Requests aufgebaut. Das Ergebnis der Prognose wird zusammen mit Informationen über das aktuelle Verkehrsaufkommen und Wetterangaben als Input für ein Neuronales Multilayer Feedforward Netzwerk verwendet. Das Neuronale Netzwerk liefert die Prognose für den Zeitbedarf von Streckenabschnitten. Die Berechnungen für die Prognosen sind bei Attanasio u. a. einigermaßen aufwendig, weswegen hier auf die Angabe der Formeln verzichtet wird.

2.4.4 Strategien zur Behandlung von dynamisch–stochastischen VRP

Die bisher zur Optimierung von dynamisch–stochastischen VRP verwendeten Algorithmen sind Abwandlungen bzw. Erweiterungen von Algorithmen zur Optimierung von statischen VRP. Im folgenden werden die populärsten Strategien vorgestellt. Da die VRP aus dem Umfeld der Arbeit in die Klassen der schwach dynamischen bis mittel dynamischen Systeme fallen, werden die Strategien zur Optimierung von stark dynamischen Systemen hier nicht aufgeführt. Die Performanz der Algorithmen wird meist in Routenlänge bzw. abgewiesenen Requests in Abhängigkeit des DOD angegeben.

2.4.4.1 Verwendung von Zeitpuffern

Bei dynamisch stochastischen VRP ist eine Optimierung der Routen, bei der die zeitliche Verfügbarkeit der Fahrzeuge maximal ausgenutzt wird, meist nicht sinnvoll, da dies die In-

tegration spontaner Requests in bestehende Routen erschwert bzw. unmöglich macht. Eine adäquate Reaktion auf sich verändernde Umweltbedingungen wie Verkehrsaufkommen und Wetterverhältnisse wird in diesem Fall ebenfalls schwierig bzw. unmöglich. Enthalten die Routen der Fahrzeuge Zeitpuffer, im VRP Bereich *Slack* genannt, kann auf dynamisch auftretende Events besser reagiert werden. Bei VRP, bei denen den Requests Zeitfenster zugeordnet sind, können (meist kleinere) Zeitpuffer auch in hochoptimierten Routen enthalten sein. Je nach Einsatzszenario bietet es sich jedoch an, zusätzlichen Slack in die Routen einzufügen, um Spielraum für die Integration spontaner Requests schaffen.

CORDEAU U. A. nennen den Zeitpuffer, der in einer PDP–Route $(0, \ldots, i_q = 2n + 1)$ ab der Ecke i enthalten ist, den *Forward Time Slack* F_i. Zur Verringerung der Dauer von Routen wird die Strategie empfohlen, die Weiterfahrt von Fahrzeugen nach Bearbeitung eines Request bei einem Kunden maximal, ðfür die Dauer des entsprechenen Forward Time Slacks, zu verzögern [vgl. Cordeau u. a., Due 2008, S.23]. der durch folgende Formel definiert ist:

$$F_i = \min_{i \leq j \leq q} \left\{ \sum_{i \leq p \leq j} v_p + (l_j - u_j) \right\} \tag{2.22}$$

wobei v_p die Wartezeit des Transportfahrzeugs bei Ecke i, l_i das Ende des Zeitfensters bei Ecke i und u_i der Zeitpunkt ist, an dem das Fahrzeug bei der Ecke v_i eintrifft. Bei der Berechnung des Timeslacks werden durch die Formel **??** die Summen der Wartezeiten bei allen möglichen Teilstrecken ab der Ecke v_i gebildet $(\{(v_i, v_{i+1}), \ldots, v_q), (v_{i+1}), v_{i+2}), \ldots, v_q), \ldots, (v_q)\})$. Zu den Summen wird jeweils die restliche verfügbare Zeit ($l_j - u_j$ in Formel 2.22) addiert. Dies ist die maximale Zeispanne, an der ein Fahrzeug später als geplant an der jeweils letzten Ecke der Teilstrecke ankommen kann, ohne dass es zu einer Verletzung des Zeitfensters der entsprechenden Ecke kommt. Der minimale Wert, der aus einer der Summen der Wartezeiten und der restlichen verfügbaren Zeit an der letzten Ecke einer Teilstrecke gebildet werden kann, definiert den Forward Time Slack F_i. Im Original des Artikels von CORDEAU U. A. wird nach der dort angegebenen Formel über alle Teilstrecken ab der ersten Ecke der Tour iteriert. Es ist jedoch anzunehmen, dass es sich hierbei um einen Flüchtigkeits– oder Druckfehler handelt.

2.4.4.2 Wartestrategien

Intuitiv betrachtet mag es am besten erscheinen, die jeweiligen Kunden schnellstmöglich anzufahren, um den vorhandenen Slack nicht unnütz zu verbrauchen. Dies bemerken auch BRANKE U. A.. In ihrer empirisch durchgeführten Studie kommen sie jedoch zu dem Schluss, dass es in bestimmten Situationen vorteilhafter sein kann, wenn Fahrzeuge eine gewisse

Zeit an “strategisch günstigen Positionen” abwarten, bevor diese zum nächsten Kunden weiterfahren. Die Wahrscheinlichkeit, dass spontane Requests vorteilhafter eingefügt werden können, steigt dadurch an [vgl. Branke u. a., 2005, S.298/1–2]. Branke u. a. führen Versuche mit einen bis drei spontanen Requests durch. Dabei zeigt sich, dass bei nur einem spontan auftretenden Request die Anwendung der Wartestrategie keinen Vorteil bringt. Bei zwei spontanen Requests ist der Gewinn bei Anwendung der Wartestrategie maximal, bei drei spontanen Request fällt der Gewinn jedoch wieder gegenüber dem Gewinn bei zwei spontanen Requests ab [vgl. Branke u. a., 2005, S.298/12]. Die Autoren bemerken, dass bei vielen spontanen Requests die Anwendung der Wartestrategie mit hoher Wahrscheinlichkeit nicht vorteilhaft ist [vgl. Branke u. a., 2005, S.298/2].

Im Zusammenhang mit Wartestrategien werden von BRANKE U. A. die Arbeiten von MITROVIĆ–MINIĆ UND LAPORTE zitiert. Mitrović–Minić und Laporte untersuchen, inwieweit sich durch Wartestrategien Routenlängen und die Anzahl der benötigten Fahrzeuge minimeren lassen. Sie zeigen, dass bei Anwendung der Wartestrategie am Anfang der Routen kürzere Routen entstehen, jedoch bei höherer Anzahl an Fahrzeugen. Bei Anwendung der Wartestrategie gegen Ende der Routen lässt sich die Anzahl an Fahrzeugen besser minimieren, bei allerdings längeren Routen. Die besten Ergebnisse werden durch eine Mischform erzielt. Die Routen werden dazu in Segmente aufgeteilt, wobei die Wartezeiten proportional zu den Servicezeiten der Kunden innerhalb eines Segments auf die Segmente verteilt wird.

Mit dieser Strategie lassen sich sowohl die Längen der Routen als auch die Anzahl der benötigten Fahrzeuge minimieren [Mitrovic-Minic:2004, zit. n. Branke u. a., 2005, S. 289/3]. ICHOUA U. A. zitieren in diesem Zusammenhang ebenfalls MITROVIĆ–MINIĆ UND LAPORTE, wobei erwähnt wird, dass die Aufteilung der Segmente anhand der geographischen Verteilung der Kunden erfolgt, wobei näher zusammenliegende Kunden jeweils in einem Segment zusammengefasst werden. Innerhalb eines Segments verlassen die Fahrzeuge die Kunden so bald wie möglich, vor dem Überqueren einer Segmentgrenze warten die Fahrzeug so lange wie möglich [Mitrovic-Minic:2004, zit. n. Ichoua u. a., 2007, S. 13].

2.4.4.3 Multiple Plan Approach

Der *Multiple Plan Approach* (MPA) wird von BENT UND HENTENRYCK als Vorgehensweise zur Handhabung von dynamisch–stochastisch auftretenden Requests vorgeschlagen [Bent und Hentenryck, 2004, vgl.]. Die Grundidee besteht darin, zu jedem Zeitpunkt während des Planungshorizonts mehrere Lösungen, Pläne genannt, für das jeweilige DSVRP beizubehalten. Aus der Menge dieser Pläne wird ein *distinguished Plan* gewählt, nach dem die Fahrzeugflotte operiert. Ein *Event–Handling* stellt sicher, dass alle Pläne mit getroffenenen

Routing–Entscheidungen und dem distinguished Plan konsistent sind. Dabei werden folgende Events behandelt:

- Customer Request: bei Auftreten eines neuen Requests wird ermittelt, in welche Pläne der neue Request eingefügt werden kann. Kann der Request in keinen Plan eingefügt werden, muss er abgelehnt werden.
- Plan Generation: wird bei der laufenden Optimierung ein neuer Plan generiert, wird er der Menge der Pläne hinzugefügt und der neue distinguished Plan wird ermittelt.
- Vehicle Departure: sieht der distinguished Plan vor, dass ein Fahrzeug von einem Kunden aus abfahren soll, werden alle Pläne gelöscht, die mit der Abfahrt inkompatibel sind.
- Timeout: wartet ein Fahrzeug v zum zeitpunkt t nach Bearbeitung eines Requests bei einem Kunden und ein Plan sieht für fahrzeug v als späteste Abfahrt bei diesem Kunden den Zeitpunkt t vor, wird der entsprechende Plan ungültig und wird gelöscht.

Die Kosten eines Plans sind als Auswahlkriterium für den distinguished Plan ungeeignet, da bei einer Optimierung des Plans bezüglich der Kosten in Routenlänge und Zeitdauer die Pläne so knapp wären, dass neu auftretende Requests nicht mehr in den Plan integriert werden könnten. Als distinguished Plan wird aus der Menge der Pläne derjenige Plan ausgewählt, der zu den anderen Plänen im Durchschnitt die höchste Ähnlichkeit aufweist. Bent und Hentenryck führen an, dass dies als *least Commitment Strategy* angesehen werden kann, da der distinguished Plan im Mittel nicht allzu sehr von den anderen Plänen abweicht [vgl. Bent und Hentenryck, 2004, S.7]. Um die Ähnlichkeit zu anderen Plänen zu berechnen, wird eine zweidimensionale Matrix M benutzt, wobei $M[v, r]$ die Anzahl der Pläne enthält, in denen Fahrzeug v den Kunden r als nächstes verlässt. Die Ähnlichkeit eines Plans σ wird durch die *Consensus Function* $f(\sigma)$ ermittelt, $f(\sigma)$ ist dabei definiert als

$$f(\sigma) = \sum_{v=1}^{m} M_t[v, succ(\sigma, \mathrm{LDC}(v))] \tag{2.23}$$

wobei σ den Plan und m die Anzahl der Fahrzeuge bezeichnet und $succ(\sigma, \mathrm{LDC}(v))$ der Nachfolger des Kunden ist, den Fahrzeug v zuletzt verlassen hat.

2.4.4.4 Multiple Scenario Approach

Der *Multiple Scenario Approach* (MSA) wird ebenfalls in Bent und Hentenryck [2004] eingeführt und ist eine Erweiterung des MPA. Beim MSA werden zukünftige Requests anhand der

stochastischen Verteilung ihres Auftretens ausgewählt und in die Pläne projiziert. Bei der Generierung von Plänen werden diese Requests den feststehenden Requests hinzugefügt und nach der Optimierung entfernt. Die Pläne enthalten somit meist genügend Spielraum, um neue auftretende Requests in die Pläne einfügen zu können. In Bent und Hentenryck [2003] führen BENT UND HENTENRYCK an, dass bei Problemen mit lockeren Randbedingungen MSA dazu tendiert, Pläne zu erzeugen, bei denen die Fahrzeuge nach Bearbeitung eines Requests sehr früh zum nächsten Kunden weiterfahren. In diesem Fall ist es vorteilhaft, die Abfahrt der Fahrzeuge zu verzögern, da in vielen Fällen dadurch die Gesamtlänge der Routen reduziert wird. Diese Strategie wird *Multiple Scenarion Approach with Least Commitment* (MSA–LC) genannt [vgl. Bent und Hentenryck, 2003, S.5]. Beim MSA wird die gleiche Consensus Function wie beim MPA verwendet, beim MSA–LC weicht die Consensus Function von der des MPA und MSA etwas ab.

2.4.4.5 LNS mit MPA und MSA

BENT UND HENTENRYCK untersuchen die Leistungsfähigkeit von LNS und der Nearest Neighbour Heuristik (NN), beide jeweils in Verbindung mit MPA bzw. MSA, bei der Anwendung auf CVRP mit Deadlines (Begrenzung für der Dauer von Routen). Als Bewertungsfunktion wird bei der Optimierung sowohl die Routenlänge (MSA^d) als auch eine *Concensus Function* (MSA^c) verwendet. Die Untersuchung erfolgt anhand von vier Modellen M1 bis M4, wobei die Modelle M1 und M2 jeweils nur ein Fahrzeug und 40 Requests beinhalten, die Modelle M3 und M4 enthalten 4 Fahrzeuge und 160 Requests, wobei einem Fahrzeug maximal 50 Requests zugeordnet werden können. Die Testinstanzen werden mit einem dod von 0% bis 100% erzeugt, wobei die Kunden stochastisch innerhalb einer quadratischen Fläche verteilt sind [vgl. Bent und Hentenryck, 2003, S.1–7]. Im folgenden werden nur die Tests an den Instanzen M3 und M4 betrachtet. Im Modell M3 sind die Kunden auf einer Fläche von 20km $\times$ 20km Gauß–verteilt, im Modell M4 sind die Kunden auf der gleichen Fläche um zwei zentralen Punkte herum Gauß–verteilt.

Die Performanz wird anhand der Anzahl an abgewiesenen Kunden und der Routenlänge – jeweils in Abhängigkeit zum dod – ermittelt. Die Testergebnisse sind als Durchschnittswerte der Performanz der Algorithmen in Abhängigkeit des dod in Diagramme eingetragen. Abbildung 2.17 und 2.18 zeigen die Diagramme für die Modelle M3 und M4.

Im Modell M3 zeigt sich die einfache Nearest Neighbour Heuristik bei der Bewertung anhand abgewiesener Requests annähernd leistungsfähig wie Nearest Neighbour in Verbindung mit MSA–LC und LNS in Verbindung mit MSA–LC. LNS mit MPA und LNS mit MPA–LC schneiden hier signifikant schlechter ab. Bei der Bewertung anhand der Routenlänge ist die Situation umgekehrt. LNS mit MSA liefert die besten Ergebnisse, gefolgt von LNS mit MPA.

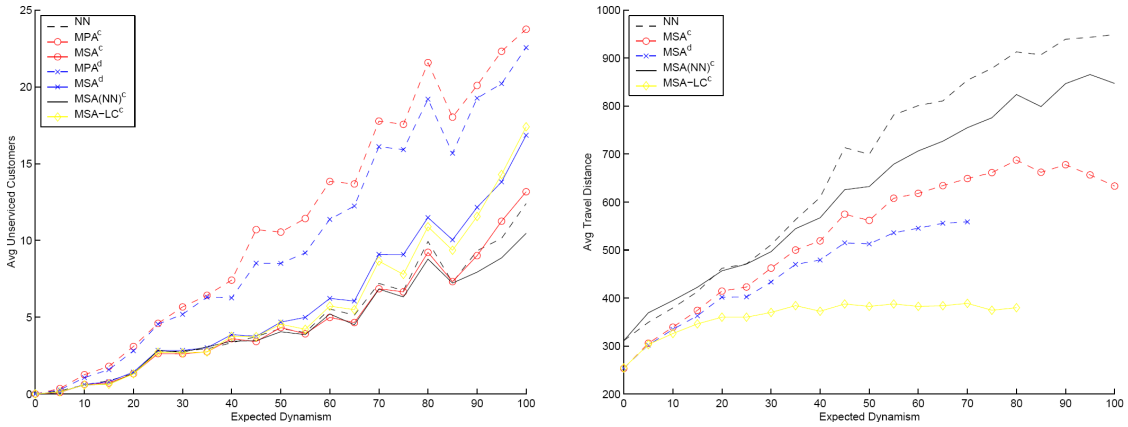

Abbildung 2.17: Abgewiesene Requests und Routenlänge bei Modell M3. Graphik aus Bent und Hentenryck [2003].

Die Ergebnisse für LNS mit MSA–LC werden hier nicht angegeben. Die Nearest neighbour Heuristik schneidet bei der Bewertung der Routenlänge am schlechtesten ab.

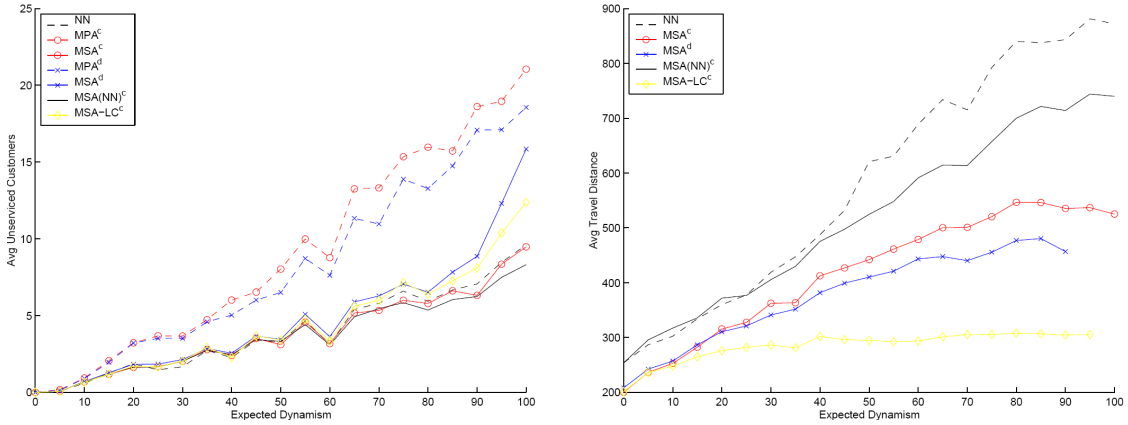

Abbildung 2.18: Abgewiesene Requests und Routenlänge bei Modell M4. Graphik aus Bent und Hentenryck [2003].

Im Modell M4 fällt die Leistung aller getesteten Algorithmen gemessen an der Anzahl abgewiesener Kunden ähnlich aus wie im Modell M3, wobei alle Algorithmen geringfügig besser abschneiden als im Modell M3. Bei der Bewertung anhand der Routenlänge schneidet die Nearest Neighbour Heuristik wieder am schlechtesten ab. LNS mit MSA ist hier um einiges besser, die Werte für LNS mit MPA sind hier nicht angegeben.

2.4.4.6 Double Horizon

ICHOUA U. A. beschreiben den *double horizon* als eine Generalisierung des *short–term rolling horizon*. Als Urheber des double horizon werden MITROVIĆ-MINIĆ U. A. angegeben. Beim short–term rolling horizon werden nur diejenigen Requests betrachtet, die in der Nähe des gegenwärtigen Zeitpunkts liegen. Beim double horizon wird sowohl ein kurzfristiger Planungshorizont (short–term planning horizon) als auch ein langfristiger Planungshorizont (long–term planning horizon) verwendet. Letzterer dient dazu, die langfristig betrachtet negativen Einflüsse von kurzfristig betrachtet guten Lösungen zu vermeiden. Beiden Horizonten sind unterschiedliche Bewertungsfunktionen zugeordnet. Beim kurzfristigen Planungshorizont werden Routen hinsichtlich ihrer Kosten bewertet, beim langfristigen Planungshorizont werden Routen bevorzugt, die mehr Slack enthalten [Mitrovic-Minic:2004b, zit. n. Ichoua u. a., 2007, S.16].

2.4.4.7 Time Slots

KILBY U. A. teilen den Planungshorizont (ein Arbeitstag) in ca. 50 Zeitintervalle, die *time slots* genannt werden. Während eines time slots läuft die Optimierung durch den Algorithmus, ein einfacher local search mit 2–Opt, Or–Opt, Node–Relocate und Node–Exchange, unterbrechnugsfrei. Am Ende des time slots wird der Algorithmus unterbrochen und die spontanen Requests, die während des time slots aufgelaufen sind, werden an der jeweils günstigsten Stelle eingefügt [vgl. Kilby u. a., 1998, S.5]. Das DSVRP wird auf diese Weise wie eine Reihe von statischen VRP behandelt. Die Unterteilung des Planungshorizonts in time slots wird auch von MONTEMANNI U. A. bei einem auf das DSVRP angepassten Ant Colony Optimization Algorithmus, Montemanni u. a. teilen einen Arbeitstag in 25 time slots auf [vgl. Montemanni u. a., 2002, S.6, 14]. GENDREAU U. A. wenden parallele Tabu Suche in Verbindung mit time slots auf das DSVRP an. Die Probleminstanzen zum Test sind dem Solomon Benchmark entnommen. Die Länge eines time slots beträgt hier 15 Minuten bei Problemen mit engen Zeitfenstern (R1, C1 und RC1) und 60 Minuten bei Problemen mit weiten Zeitfenstern (R2, C2, RC2) [vgl. Gendreau u. a., 1999, S.6].

2.5 Ansätze für interaktive Verfahren

In der wissenschaftlichen Literatur zu Vehicle Routing Problemen finden sich wenig Ansätze zu interaktiven Verfahren zur Lösung von VRP, d. h. Verfahren bei denen ein menschlicher Planer in den Optimierungsprozess eingreifen kann, um z. B. Lösungen zu verändern oder

Lösungsbestandteile zu “fixieren” und von der weiteren Optimierung auszuschließen. Kommerziell verfügbare Systeme zur Routenplanung bieten oft die Möglichkeit, auf einem Display dargestellte Routen per Drag and Drop zu verändern. In einer von HALL UND PARTYKA kürzlich durchgeführten Umfrage über Software zur Routenplanung gaben 14 von 18 Softwareherstellern an, dass ihr System die Möglichkeit bietet, Routen per Drag and Drop zu verändern [vgl. Hall und Partyka, 2008]. Informationen über die in kommerziellen Systemen verwendeten interaktiven Verfahren sind, wohl zur Wahrung von Wettbewerbsvorteilen, so gut wie nicht verfügbar.

2.5.1 Gründe für die Anwendung interaktiver Verfahren

Für die Integration von Interaktionsmöglichkeiten für menschliche Planer bestehen insbesondere bei Systemen zur Behandlung realer VRP einige gewichtige Gründe, die im folgenden aufgeführt sind:

1. ANDERSON U. A. bemerken, dass bisweilen nicht alle Randbedingungen im Modell enthalten sind [vgl. Anderson u. a., 2000, S.3]. KOPFER UND SCHÖNBERGER führen an, dass bei einigen realen Problemen nicht alle Randbedingungen formalisierbar sind [vgl. Kopfer und Schonberger, 2002, S.2].

2. Sowohl SCOTT U. A. als auch KOPFER UND SCHÖNBERGER führen an, dass menschliche Planer Computern hinsichtlich der Verarbeitung spatialer Daten und strategischer Einschätzung zum gegenwärtigen Zeitpunkt überlegen sind. KOPFER hebt in diesem Zusammenhang die Erkennung unstabiler Pläne hervor, bei denen die Qualität der Lösung von einigen wenigen Bestandteilen der Lösung abhängt [vgl. Scott u. a., 2002, S.2] und [vgl. Kopfer und Schonberger, 2002, S.4, 7].

3. SCOTT U. A. heben hervor, dass menschliche Planer eine generierte Lösung verstehen und ihr Vertrauen müssen. Dies wird als wichtige Grundlage für die Implementation eines Plans angesehen [vgl. Scott u. a., 2002, S.2].

4. Reale Probleme sind meist anhand mehrerer, oft zueinander in Konkurrenz stehenden Kriterien zu optimieren. Die Bewertungsfunktion wertet Lösungen anhand der Gewichtung der einzelnen Kriterien aus. Nach KOPFER UND SCHÖNBERGER sollte eine Kontrolle der Ausbalanzierung der Gewichtung durch den menschlichen Planer erfolgen, um die Dominanz oder Unterbewertung einzelner Kriterien zu vermeiden [vgl. Kopfer und Schonberger, 2002, S.6].

5. KOPFER UND SCHÖNBERGER definieren ein *schlecht strukturiertes Problem* als ein Problem, dessen Struktur dazu führt, dass das Verfahren zur Lösung des Problems vorzugsweise diejenigen Teile des Lösungsraum ineffizient durchsucht, welche keine

guten Lösungen bieten. Unglücklicherweise sind solche Probleme in den meisten Fällen nicht im vorhinein zu erkennen. Durch eine Überwachung des Lösungsprozesses können diese Situationen erkannt werden. Ein menschlicher Planer kann dann durch geeignete Interaktion das Verfahren in Regionen des Lösungsraums leiten, in denen bessere Lösungen zu erwarten sind [vgl. Kopfer und Schonberger, 2002, S.5].

6. Bei der Optimierung von realen dynamisch–stochastischen VRP können Events auftreten, die die Interaktion eines menschlichen Planers erforderlich machen. Dazu gehören u. a. der Ausfall von Fahrzeugen während einer Tour oder Blockierung von Verkehrsverbindungen. KOPFER UND SCHÖNBERGER sehen bei kurzfristigen Ausfällen von Fahrzeugen oder kurzfristigen Verkehrsblockaden die Möglichkeit der Vergabe von entsprechend kurzfristig geltenden Randbedingungen [vgl. Kopfer und Schonberger, 2002, S.6].

2.5.2 Human Guided Simple Search

Bei den Mitsubishi Research Labs untersucht eine Gruppe um David Anderson, Gunnar Klau, Stacey D. Scott u. a. Möglichkeiten der Interaktion beim Lösen von dynamisch–stochastischen VRP. ANDERSON U. A. entwickeln das *Human Guided Simple Search* (HuGSS) als Plattform für die Untersuchungen. Die folgenden Informationen wurden, soweit nicht anderweitig gekennzeichnet, aus [Anderson u. a., 2000] entnommen.

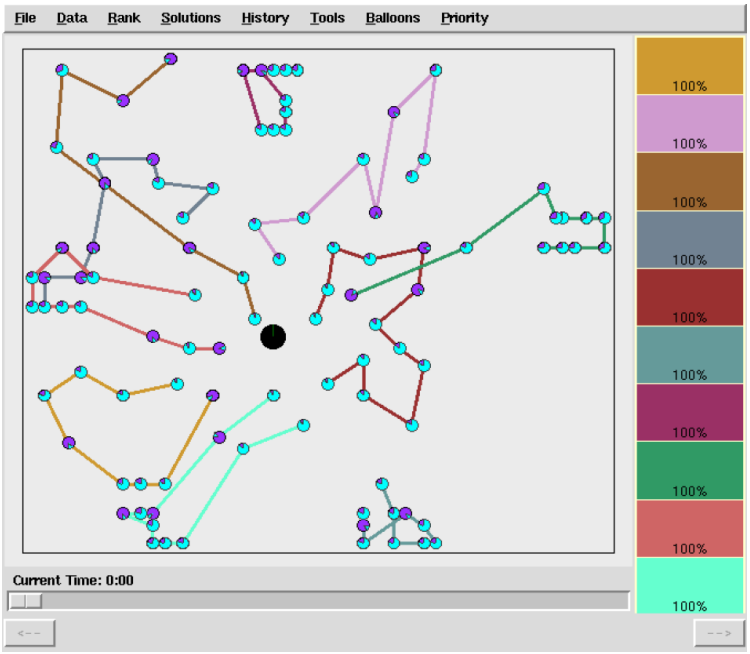

Abbildung 2.19: User Interface des HuGSS. Graphik aus **?**.

Das System besteht aus einem einfachen Hillclimbing Algorithmus, einem Display zur Anzeige von Lösungen des VRP und bietet folgende Möglichkeiten der Interaktion:

1. Die Verschiebung eines Requests von einer Route in eine andere Route. Dies ist als *1–Ply Move* definiert.
2. Die Vergabe von Mobilitäts–Prioritäten, *Mobility* genannt, wird von KLAU U. A. beschrieben. Jedem Request kann die Mobility *high*, *medium* oder *low* zugeordnet werden. Zwei benachbarte Requests können getauscht werden, wenn mindestens einer der Requests die Mobility 'high' besitzt und keiner der beiden Requests die Mobility 'low' besitzt. Wird einem Request die Mobility 'low' zugeteilt, wird die Route, in der sich der Request befindet, in zwei Subprobleme aufgeteilt [vgl. Klau u. a., 2002, S.5]. Abbildung 2.20 veranschaulicht den Zusammenhang.
3. Start der lokalen Suche ausgehend von der gegenwärtigen Lösung. Dabei kann vom menschlichen Planer ausgewählt werden, welche maximale Anzahl n von Requests von einer Route in eine andere Route vom Algorithmus verschoben werden können, wobei $n = \{1, \ldots, 5\}$. Dies ist als *n–Ply Move* definiert.
4. Die Auswahl einer vorhergehenden Lösung als Startpunkt für die weitere Suche.

Nach einem 1–Ply Move werden die beiden betroffenen Routen des VRP vom System durch 1–Ply–Moves bis n–Ply–Moves optimiert, wobei n durch den Dispatcher festgelegt wird. In jeder Iteration des Hillclimbing Algorithmus werden die n–Ply Moves auf die betreffenden Routen des VRP angewendet. Für den Hillclimbing Algorithmus des Systems ist eine der beiden Strategien *greedy Mode* und *steepest Descent Mode* auszuwählen. Beim greedy Mode wird nach der ersten Verbesserung einer Route eine neue Iteration gestartet, beim steepest Descent Mode wird nur der Veränderungsschritt der n–Ply Moves akzeptiert, der die höchste Einsparung an Kosten bewirkt. Der Einfluss der Mobilities beim Austausch von Requests ist nicht beschrieben.

SCOTT U. A. untersuchen in Experimenten mit dem HuGSS die Leistungsfähigkeit menschlicher Planer. Die Autoren erwähnen ausdrücklich, dass die Experimente nicht zum Ziel haben, zu beweisen, dass menschliche Planer bei der Optimierung von VRP bessere Ergebnisse erzielen als automatische Systeme. Dies würde nach Ansicht der Autoren erfordern, bei einem automatischen Algorithmus alle möglichen Arten von Einstellungen (z. B. hinsichtlich der Mobilities) zu testen. [vgl. Scott u. a., 2002, S.6]

An den Experimenten nehmen drei Versuchspersonen teil, die alle graduierte Studenten bzw. professionelle Softwareentwickler sind. Alle drei Versuchspersonen hatten durch eine vorherige Studie mindestens acht Stunden mit dem HuGSS gearbeitet und dadurch etwas Erfahrung. Die verwendeten VRP stammen aus dem Solomon Benchmark, allerdings wird nicht angegeben, welche Testinstanzen daraus verwendet werden. Die Tests haben die zwei folgenden Schwerpunkte:

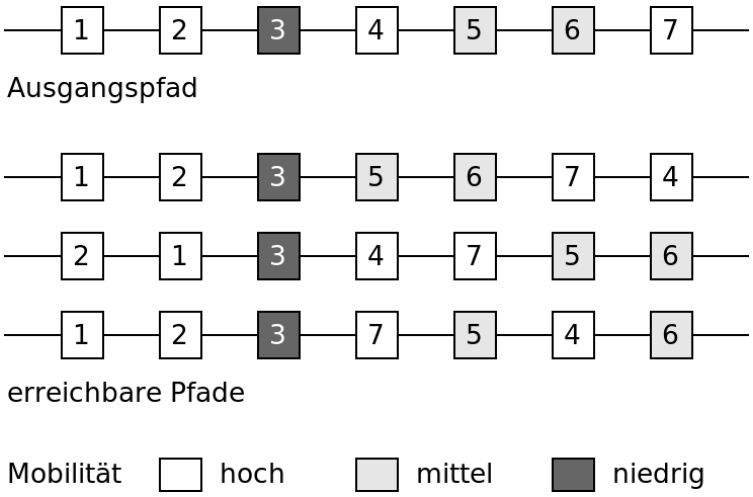

Abbildung 2.20: Steuerung der Permutation von Requests durch Mobilities. Graphik aus Anderson u. a. [2000].

1. Identifizierung günstiger Teilbereiche des Lösungsraums. Durch die Vergabe von Mobilities soll die Optimierung des automatischen Teils des HuGSS auf möglichst günstige Teilbereiche des Lösungsraums fokussiert werden.
2. Abbruch des lokalen Suchalgorithmus. Nach Veränderung einer Lösung durch den menschlichen Planer wird diese maximal zwei Minuten durch den lokalen Suchalgorithmus optimiert. Der menschliche Planer soll den Algorithmus vorher abbrechen, wenn erwartet wird, das in der verbleibenden Laufzeit keine nennenswerten Verbesserungen erzielt werden.

SCOTT U. A. kommen aufgrund der Ergebnisse zu dem Schluss, dass die drei Versuchspersonen hinsichtlich der zwei Kriterien sehr gut abschneiden. Durch die Vergabe von Mobilities können die menschlichen Planer 63.3% der Lösungen, die sich durch den lokalen Suchalgorithmus ergeben, verbessern (zwei Personen erzielen 70%, eine Person erzielt 50%). Beobachtungen zeigen, dass vor allem in den ersten 30 bis 40 Sekunden der Optimierung durch die Vergabe von Mobilities durch menschliche Planer Leistungssteigerungen erzielen lassen. Bei der Auswahl des Zeitpunkts zum Abbruch des Algorithmus werden die Ergebnisse bei Abbruch mit den Ergebnissen der maximalen Laufzeit von zwei Minuten verglichen. Die Versuchspersonen brechen die Optimierungsläufe durchschnittlich nach einer Minute ab. Zwei Testpersonen erzielen dabei knapp über 80% der Verbesserungen gegenüber zwei Minuten Laufzeit, eine Testperson erzielt 73.8% der Verbesserungen gegenüber zwei Minuten Laufzeit. Die Verbesserungen bei den vollen zwei Minuten Laufzeit betragen nach 90 Sekunden 80%, bei gleicher Qualität der Lösungen werden von den Versuchspersonen also ca. 30 Sekunden “gespart”. Die Autoren kommen zu dem Schluss, dass die Versuchspersonen bei

der Wahl des Zeitpunkts zum Abbruch sehr gut einschätzen können, wann ein Großteil der möglichen Verbesserungen erreicht ist [vgl. Scott u. a., 2002, S.6–8].

3 Analyse

In diesem Kapitel erfolgt eine Analyse der Problemstellung sowie der Randbedingungen der Problemstellung. Da die Optimerung des jeweiligen zu optimierenden Systems sowohl durch einen Optimierungsalgorithmus als auch interaktiv durch einen Dispatcher erfolgen soll, besteht sowohl die Frage, welche Interaktionsmöglichkeiten für einen Dispatcher notwendig bzw. sinnvoll sind, als auch die Frage nach dem Zusammenspiel von automatischer Optimierung durch den Algorithmus und interaktiver Optimierung durch den Dispatcher. Aufgrund dieser Fragestellung erfolgt im Anschluss an die Analyse der Problemstellung und deren Randbedingungen zunächst eine Analyse des Problems in Hinsicht auf eine interaktive Optimierung durch einen Dispatcher. Darauf folgend werden die im zweiten Kapitel (Grundlagen) dargestellten Lösungsansätze und Verfahren auf ihre Verwendbarkeit hinsichtlich des Problems unter Berücksichtigung der Randbedingungen sowie der Interaktionsmöglichkeiten untersucht.

3.1 Analyse der Problemstellung

Bei der Analyse der Problemstellung sind zunächst folgende Aspekte zu beachten: das Kundenprofil, die räumlichen Verteilung der Kunden, die Art der Aufträge zur Belieferung und Abholung von Werten an bzw. von Kunden sowie die Anzahl von *a-priori* und spontanen Anfragen. Des weiteren besteht die Frage nach der Kapazität der Transportfahrzeuge und der durchschnittlichen Anzahl der Kunden, die pro Tour eines Fahrzeugs beliefert werden. Da keine aussagekräftige Daten von realen Einsatzszenarios zur Verfügung stehen, muss hier eine Abschätzung erfolgen. Werden dabei verschiedene (angenommene) Einsatzszenarios betrachtet, bei denen die Ausprägung bestimmter Randbedingungen wie z. B. Zeitfenster signifikant voneinander abweichen, stellt dies keinen Nachteil dar, da dadurch die Robustheit des Optimierungsverfahrens gegenüber der Eingabe von Probleminstanzen aus einem weiten Spektrum an möglichen realen Einsatzszenarios mit unterschiedlichen Ausprägungen in den einzelnen Randbedingngen getestet werden kann. Da die Anbindung des zu erstellenden Systems an ein Geoinformationssystem nicht Bestandteil dieser Arbeit ist, können die geographischen Standorte der Kunden nicht bestimmt werden. Aufgrund dieser Einschränkung wird der Analyse eine Verteilung der Kunden zugrundegelegt, bei der Kunden sowohl zufallsverteilt als auch in *Clustern* verteilt sind. Ein Cluster bezeichnet dabei eine Anzahl

von Kunden, die in einem bestimmten Gebiet dichter zusammenliegen als Kunden mit zufallsverteilten Standorten. Diese Art der Verteilung (zufallsverteilt und in Clustern) entspricht weitgehend der Verteilung der Kundenarten wie z. B. Supermärkte und Banken in “natürlich gewachsenen” städtischen Gebieten.

Anfragen bzw. Aufträge von Kunden werden im folgenden mit *Requests* bezeichnet, wobei Anfragen zur Belieferung eines Kunden vom Depot aus als *Delivery–Request*, Anfragen zur Abholung von Werten vom Kunden mit Abgabe der Werte an das Depot als *Pickup–Request* und Anfragen zum Transport von Werten von einem Kunden zu einem anderen Kunden als *Pickup–und–Delivery–Request* bzw. mit der Abkürzung *PD–Request* bezeichnet werden. *A–priori* Requests sind Requests, die vor Beginn der *a–priori* Optimierung bekannt sind. Spontane Requests sind diejenigen Requests, die während des Planungshorizonts bekannt werden. Als *Stop* wird im folgenden eine Entität bezeichnet, die den Ursprungs– oder Bestimmungsort beim Transport von Werten darstellen kann, d. h. Kunden und das Depot. Als *Tourstop* wird im folgenden ein Stop bezeichnet, der an einem Request als Ursprungs– oder Bestimmungsort beim Transport von Werten beteiligt ist. Dadurch ist es möglich, während eines Planungshorizonts einem Stop mehrere Aufträge zuzuteilen [1].

Zwei besondere Randbedingung sind die sich aus der Art der Unternehmung bedingende Art der Abfertigung von Transportfahrzeugen im Depot und die Aufteilung der Kundenprofile in Klassen, wobei eine Klasse von Kunden (Kunden der Art “Geldautomat”) nur von Fahrzeugen mit drei Besatzungsmitgliedern bedient werden kann, während die andere Klasse von Kunden (alle Kunden außer denen der Art “Geldautomat”) von Fahrzeugen mit mindestens zwei Besatzungsmitgliedern bedient werden kann.

Der Planungshorizont wird bei allen Instanzen als gleich angenommen und liegt im Zeitraum von 08:00 Uhr bis 20:00 Uhr.

3.1.1 Kundenprofil

Die Tabelle 3.1.1 zeigt die angenommene Verteilung von Kundenarten, anhand derer Instanzen zum Test des Systems erzeugt werden.

3.1.2 Fahrzeugkapazität

Die Menge von Werten, die durchschnittlich bei den jeweiligen Kunden auszuliefern bzw. von diesen abzuholen ist, unterliegt sowohl aus sicherheitstechnischen Gründen als auch als Be-

[1] Die Aufteilung in Tourstops und Stops stellt in gewissen Sinne einen Vorgriff auf das Design des Systems dar. Die Einführung dieser beiden Begriffe an dieser Stelle ist jedoch für weitere Betrachtungen bei der Analyse nötig

Kundenart	Request	Zeitfenster	Servicezeit in Minuten	Kapazität	Prozentualer Anteil a–priori
Bank	Delivery	09:00 – 16:00	9	10	10
Geldautomat (ATM)	Delivery	08:00 – 18:00	20	10	10
Geldautomat, Wartung	–	08:00 – 20:00	30	0	5
Grosshändler	Pickup	10:00 – 18:00	20	5	25
Einzelhändler	Pickup	17:00 – 20:00	5	1	50

Tabelle 3.1: Kunden–Arten

triebsgeheimnis des Kunden der Geheimhaltung. Auch die Obergrenze an Werten, die mit einem Fahrzeug transportiert werden kann und versicherungstechnisch bedingt ist, unterliegt dem Betriebsgeheimnis des Werttransporteurs. Aufgrund dieser Beschränkung wird die Kapazität der Transportfahrzeuge mit 100 festgesetzt. Die durchschnittliche Fahrzeugkapazität, die zur Abwicklung eines Requests einer Kundenart nötig ist, wird mit einer Zahl zwischen 1 und 100 angegeben. Dies ermöglicht die Modellierung des Problems und die Erzeugung von Testinstanzen zum Test des Optimierungsverfahrens.

3.1.3 Sicherheitsschleusen des Depots

Die Depots des Werttransporteurs sind mit Sicherheitsschleusen ausgestattet, die von den Transportfahrzeugen beim Verlassen des Depot und bei der Anfahrt zum Depot zu passieren sind. Die Anzahl der Schleusen, bzw. die Anzahl der Fahrzeuge, die in einer Schleuse gleichzeitig abgefertigt werden können, können bei verschiedenen Depot jeweilis unterschiedlich sein. In den Sicherheitsvorschriften des BGV C7 [vgl. u. a. Paulick, 2007] sind keine Angaben zu verbindlichen Richtlinien über die Ausgestaltung der Sicherheitsschleusen zu finden. Bei einer freien Konfigurierbarkeit des Systems hinsichtlich der Anzahl der Schleusen und der Aufnamhekapazität der Schleusen ist das Fehlen dieser Information jedoch unkritisch. Die Zeit, die zur Abfertigung eines Fahrzeugs in der Schleuse benötigt wird, kann als auftragsabhängig angenommen werden. Da die Bearbeitungszeiten erst nach der Erstellung einer Tour berechnet werden können, sollten hier Durchschnittswerte angenommen werden. Da die durchschnittlichen Abfertigungszeiten jeweils beim Verlassen des Depots bzw. bei der Anfahrt zum Depot voneinander abweichen können, sollte bei der Defintion von die Konfiguration von zwei Werten, der durchschnittlichen Bearbeitungszeit beim Verlassen des Depots sowie bei der Anfahrt zum Depot möglich sein. Angenommen werden hier 10 Minuten beim Verlassen des Depots und 20 Minuten bei der Anfahrt zum Depot. Eine wichtige Randbedingung besteht darin, dass Fahrzeuge bei der Anfahrt zum Depot die Schleuse möglichst ohne Wartezeit bzw. nur mit geringer Wartezeit befahren können. Stauungen vor dem Depot sollten vermieden werden.

3.2 Analyse der Interaktionsmöglichkeiten

Bei der Analyse möglicher Interaktionen durch den Dispatcher können zunächst drei Phasen unterschieden werden, in denen der Dispatcher interaktiv an der Optimierung des jeweiligen Problems mitwirken kann:

- Phase 1: A–priori Optimierung des Problems durch den Optimierungsalgorithmus des Systems. Benötigt der Optimierungsalgorithmus eine Startlösung, lässt sich diese Phase in zwei weitere Phasen unterteilen: Generierung der Startlösung, z. B. durch eine Konstruktionsheuristik (Phase 1.1) und nachfolgende Optimierung durch den Optimierungsalgorithmus (Phase 1.2).
- Phase 2: Diese Phase schließt den Zeitraum vom Beenden bzw. der Terminierung des Optimierungsalgorithmus bis zum Start des Planungshorizonts ein.
- Phase 3: Optimierung während des Ablaufs des Planungshorizonts. Dies beinhaltet die weitere Optimierung des Problems während des Planungshorizonts, gegebenfalls aufgrund veränderter Bedingungen wie z. B. Fahrzeit zwischen zwei Punkten, sowie das Routen spontaner Requests.

Eine Interaktion des Dispatchers während der Generierung von Startlösungen (Phase 1.1) erscheint wenig sinnvoll. Das Zusammenstellen von Tourstops zu Touren erfordert umfangreiche Berechnungen zur Überprüfung der Einhaltung der Randbedingungen beim Erzeugen einer Tour. Durch die hohe Geschwindigkeit von Computern beim Verarbeiten numerischer Daten sind diese hier einem menschlichen Dispatcher gegenüber im Vorteil. Die automatische Erstellung einer Startlösung, gegenenfalls mit Nachbearbeitung der Startlösung durch den Dispatcher, erscheint hier sinnvoller.

Während der *a–priori* Optimierung durch den Optimierungsalgorithmus (Phase 1 bzw. Phase 1.2 bei Systemen, bei denen eine Konstruktionsheuristik zum Einsatz kommt) kann eine Arbeitsteilung zwischen Computer und Dispatcher erfolgen. Beim Durchsuchen von Bereichen des Lösungsraums nach lokalen Optima ist die hohe Geschwindigkeit des Computers bei der Verarbeitung numerischen Daten vorteilhaft. Das Durchsuchen von Bereichen des Lösungsraums nach der besten Lösung innerhalb des jeweiligen Bereichs kann also hauptsächlich dem Optimierungsalgorithmus überlassen werden. Bei der Erfassung und Verarbeitung spatialer Daten sind Menschen, insbesondere erfahrene Dispatcher, zum gegenwärtigen Zeitpunkt Computersystemen gegenüber überlegen. Ein Beispiel dafür aus dem Bereich der Problemstellung ist eine Zuordnung von Tourstops zu Touren, die zu Touren mit vermeidbaren hohen Kosten führen. Ein menschlicher Dispatcher wird in solchen Fällen schnell die Bestandteile einer Lösung bzw. eines Plans sehen, bei denen eine Modifikation eine Verringerung der Kosten der Lösung bzw. des Plans bewirkt. Der Dispatcher wird dabei aber in

vielen Fällen nicht oder nur sehr schwer in der Lage sein, sein Vorgehen formal zu beschreiben. Besteht in dieser Situation für den Dispatcher die Möglichkeit, die Lösung interaktiv zu verändern, kann dieser durch Modifikation der augenblicklichen Lösung den Optimierungsalgorithmus aus einem lokalen Optimum befreien und in einen Bereich des Lösungsraum führen, in dem bessere Lösungen zu erwarten sind, als dies für den gegenwärtigen Bereich der Fall ist. Sinnvoll erscheint hier ein Zusammenspiel von Optimierungsalgorithmus und Dispatcher, bei dem der Dispatcher die automatische Optimierung der Probleminstanz mitverfolgen kann, wobei ihm die Möglichkeit gegeben ist, den Optimierungsalgorithmus zu einem beliebigen Zeitpunkt zu unterbrechen, um die gegenwärtige Lösung selbst zu modifizieren.

Im Zeitraum nach der Terminierung des Optimierungsalgorithmus und vor Beginn des Planungshorizonts (Phase 2) kann durch den Dispatcher die Lösung weiter optimiert werden, soweit dafür Ansatzpunkte vorhanden sind.

Bevor ein Plan während des Planungshorizonts zur Ausführung kommen kann, muss dieser durch einen Dispatcher geprüft und bestätigt werden. Dies ist inbesondere bei der Art der Unternehmung von Werttransporteuren wichtig, bei denen der Aspekt der Sicherheit von Transportwegen eine besondere Rolle spielt. Dies wird auch aus den Sicherheitsvorschriften des BDGW deutlich. So sind z. B. Abweichungen vom geplanten Tourenverlauf nur nach vorheriger Absprache mit der Einsatzzentrale erlaubt. [vgl. Paulick, 2007, S.3]. Eine weitere automatische Optimierung vorhandener Touren würde den Dispatcher mit Zwangssituationen konfrontieren, in denen eine Abnahme bzw. Ablehnung der neuen gegenwärtigen Lösung von diesem zu erfolgen hat. Der erhöhte Aufwand an Kommunikation zwischen Einsatzzentrale und Transportfahrzeugen kann den ordnungsgemäßen Ablauf der Abarbeitung von Touren erschweren. Spontane Anfragen (z. B. Anfragen von Supermärkten nach Wechselgeld) sollten ebenfalls vom Dispatcher behandelt werden, da dieser situationsbedingt zu entscheiden hat. Die jeweils entsprechenden Einflussfaktoren, die für die jeweilige Entscheidung mitbestimmend sind, sind dabei oft nicht formalisierbar. Das Subsystem zur automatischen Optimierung könnte hier dem Dispatcher nur Vorschläge zum Routen spontaner Requests liefern. Eine automatische Optimierung während des Planungshorizonts (Phase 3) ist daher wenig sinnvoll.

Bei Systemen, bei denen der Optimierungsalgorithmus grundsätzlich nicht terminiert, sondern beim Übergang von der *a–priori*–Phase zum Planungshorizont und während des Planungshorizonts durchläuft, fällt Phase 2 weg. Systeme mit rollendem Planungshorizont, d. h. einem unendlichen Planungshorizont mit rollendem Zeitfenster des Depots, werden hier nicht betrachtet.

3.2.1 Interaktive Veränderung von Plänen

Als Plan wird im Folgenden eine Lösung des Optimierungsproblems bezeichnet, die sich in folgende Bestandteile unterteilen lässt:

- Das Depot.
- Eine Menge von Touren und die in den Touren gerouteten Tourstops sowie die den Touren zugeordneten Fahrzeuge.
- Eine Menge von Tourstops, die in keiner Tour geroutet sind.
- Eine Menge frei verfügbarer Fahrzeuge, d. h. Fahrzeuge, die keiner Tour zugeordnet sind.

Bei der interaktiven Veränderung von Plänen können zwei Arten der Modifikation durch Interaktion unterschieden werden: die direkte Modifikation, z. B. durch Entfernung und Wiedereinfügen eines Tourstops an anderer Stelle, und die indirekte Modifikation durch Änderung von Parametern des Optimierungsalgorithmus, z. B. das Setzen von Schwellwerten, bei denen zusätzliche Zeit–Slacks in Touren eingefügt werden. Eine Berücksichtigung beider Interaktionsmöglichkeiten ist im Rahmen dieser Arbeit nicht möglich. Die direkte Interaktion entspricht mehr dem üblichen Vorgehen eines Dispatchers bei der Erstellung von Touren und ermöglicht situationsbedingt zielgerichtetes Eingreifen des Dispatchers bei der Optimierung von Plänen. Deswegen wird im Folgenden nur die direkte Interaktion behandelt. Dabei ergeben sich folgende mögliche Basis–Operationen zur Interaktion durch den Dispatcher:

- Routen eines bisher ungerouteten Tourstops.
- Entfernung eines gerouteten Tourstops aus einer Tour.
- Austausch zweier gerouteter Tourstops. Dabei wird ein Tourstop A aus einer Tour entnommen und ein Tourstop B in die ursprüngliche Position des Tourstops A geroutet. Tourstop A wird in die ursprüngliche Position des Tourstops B geroutet. Die beiden Tourstops können dabei sowohl in der gleichen als auch in zwei unterschiedlichen Touren geroutet sein.
- Erzeugung einer neuen, leeren Tour, soweit die Menge an verfügbaren Fahrzeugen nicht leer ist. Als leere Tour wird eine Tour bezeichnet, in die zunächst keine Tourstops geroutet sind.

Diese Operatoren entsprechen weitgehend den im Kapitel Grundlagen vorgestellten Modifikationsoperatoren. Weitere Operationen wie z. B. das Verschieben eines Tourstops aus einer Tour in eine andere Tour oder die Auflösung einer Tour durch Entfernung aller Tourstops aus dieser Tour und Freistellung des dieser Tour zugeordneten Fahrzeugs zur weiteren Verwendung in einer anderen, neuen Tour, können von diesen Basis–Operationen abgeleitet

werden. Wird durch die Durchführung einer der oben aufgeführten Basis–Operationen ein Tourstop eines PD–Requests aus einer Tour entfernt oder in eine Tour geroutet, muss sichergestellt werden, dass der zugehörige zweite Tourstop ebenfalls entfernt bzw. in die gleiche Tour wie der erste Tourstop geroutet wird. Zusätzlich dazu muss sichergestellt sein, dass der Delivery–Tourstop des PD–Requests nur nach dem Pickup–Tourstop in der Tour enthalten ist. Durch Anwendung des von den Datenbanken her bekannten Transaktionsprinzips lässt sich dies erreichen. Die entsprechenden, durch den Dispatcher durchzuführenden Aktionen können hier dadurch erzwungen werden, dass das System für alle anderen Aktionen des Dispatchers gesperrt ist. Dem Dispatcher wird dabei jedoch die Möglichkeit gegeben, das Routen eines PD–Requests in eine Tour oder das Entfernen eines PD–Requests aus einer Tour abzubrechen, wobei vom System diejenigen Aktionen des Dispatchers rückgängig gemacht werden, die die Tourstops des entsprechenden PD–Requests betreffen.

Der Austausch eines PD–Requests mit einem anderen Request erscheint als Interaktionsmöglichkeit wenig sinnvoll. Die Benutzerführung wäre in diesem Fall komplex und dadurch für den Dispatcher schnell verwirrend.

3.2.2 Erzeugung ungültiger Lösungen

Grundsätzlich stellt sich die Frage, ob es dem Dispatcher möglich sein soll, bei der Veränderung von Plänen ungültige Pläne, wie z. B. Pläne mit Verletzung von Zeitfenstern, erzeugen zu können. Mögliche Gründe für die Einführung dieser Möglichkeit sind die (einfachere) Erzeugung von gültigen Plänen über Zwischenschritte, die ungültige Pläne darstellen, sowie die Behandlung von Ausnahmesituationen. Eine denkbare Ausnahmesituation wäre ein spontaner Request, dessen Bearbeitung dringend ist, jedoch aufgrund des späten Auftretens des Requests zur Folge hat, dass das Fahrzeug zur Bearbeitung des Requests nur verspätet, d. h. nach dem letzten Zeitpunkt des Zeitfensters des Depots zum Depot zurückkehren kann.

Eine Einführung der Möglichkeit zur Erzeugung ungültiger Pläne bedingt jedoch mehrere Vorraussetzungen. Zum einen sollte der eingesetzte Optimierungsalgorithmus in der Lage sein, ungültige Lösungen zu bearbeiten und diese möglichst in eine gültige Lösung überführen zu können. Dies kann bei stochastischen Algorithmen jedoch nicht garantiert werden. Zum anderen sollten in der Darstellung von Plänen in der Benutzerschnittstelle ungültige Lösungsbestandteile für den Dispatcher klar erkennbar gekennzeichnet sein. Die Benutzerschnittstelle wird jedoch damit um einiges komplexer. Die Erstellung eines Systems mit der Möglichkeit zur Erzeugung ungültiger Lösungen durch Interaktion des Dispatchers wird deswegen im Rahmen dieser Arbeit als nicht sinnvoll betrachtet.

3.3 Anwendbarkeit der bekannten Verfahren auf die Problemstellung

3.3.1 Analyse der Konstruktionsheuristiken

3.3.1.1 Der Clarke und Wright Savings Algorithmus

Aufgrund der Randbedingung der Bearbeitungszeiten der Fahrzeuge bei der Abfahrt vom bzw. der Ankunft beim Depot ist der Clarke und Wright Savings Algorithmus kaum anwendbar. Hier besteht folgendes Dilemma: Bei n Tourstops sind zunächst n Touren mit n Fahrzeugen zu erzeugen. Werden zuerst Touren erzeugt, denen die entsprechenden Zeitfenster zur Bearbeitung der Fahrzeuge bei der Abfahrt vom bzw. der Ankunft beim Depot zugeteilt werden, ist die Anzahl der verfügbaren Zeitfenster schnell erschöpft und wird bei Instanzen der Problemstellung der Arbeit nicht ausreichend sein. Werden andererseits den Touren die Zeitfenster zur Bearbeitung der Fahrzeuge in der Depotschleuse erst nach der Terminierung des Konstruktionsalgorithmus zugewiesen, besteht die hohe Wahrscheinlichkeit, dass nicht jeder Tour Zeitfenster zur Bearbeitung der Farzeuge in der Depotschleuse zugewiesen werden können, ohne dass es zu Verletzungen von Zeitfenstern bei der Anfahrt von Tourstops oder alternativ zu Kollisionen bei der Vergabe von Zeitfenstern für die Bearbeitung der Fahrzeuge in den Depotschleusen kommt – in beiden Fällen wäre die Lösung ungültig. Ein Mittelweg, bei dem iterativ jeweils einige Touren erzeugt werden, denen aus den noch verbleibenden Zeitfenstern zur Bearbeitung der Fahrzeuge in der Depotschleuse jeweils die günstigsten Zeitfenster zugeteilt werden, erscheint ebenfalls wenig aussichtsvoll.

3.3.1.2 Der Sweep Algorithmus

Die eben genannte Problematik ist beim Sweep Algorithmus nicht vorhanden, da hier nacheinander Touren erzeugt werden, denen solange Tourstops zugeordnet werden, bis eine Randbedingung wie die Fahrzeugkapazität oder die maximale Tourlänge verletzt wird. Somit erscheint dieser Konstruktions–Algorithmus für die Anwendung auf die Problemstellung zunächst als geeignet. Allerdings muss bei der Zuordnung der Tourstops zu Touren darauf geachtet werden, dass diese nur den Touren zugeordnet werden, die von Fahrzeugen abgefahren werden, die über die für die Abwicklung des Requests benötigte Mindestbesatzungsstärke aufweisen.

3.3.2 Analyse der Modifikationsoperatoren

Die im Kapitel Grundlagen vorgestellten Modifikationsoperatoren sind für die Problemstellung einsetzbar, soweit bei der Anwendung auf Tourstops von PD–Requests zusätzlich zu den auch für Tourstops von Pickup– und Delivery–Request bestehenden Bedingungen die Bedingungen eingehalten werden, die schon bei der Modifikation eines Plans durch einen Dispatcher genannt wurden (beide Tourstops müssen in der gleichen Tour enthalten sein und der Delivery–Tourstop darf nur nach dem Pickup–Tourstop auftreten).

3.3.3 Analyse der Metaheuristischen Verfahren

Bevor eine Bewertung der im Kapitel Grundlagen vorgestellten msetaheuristischen Verfahren erfolgen kann, müssen aufgrund der Problemstellung mit der Forderung nach Möglichkeiten zur Interaktion durch den Dispatcher einige grundsätzliche Überlegungen zu den Auswahlkriterien angestellt werden. Zur Beurteilung eines metaheuristischen Algorithmus können im allgemeinen folgende Kriterien herangezogen werden:

1. Einfachheit: der Algorithmus sollte möglichst einfach zu implementieren sein.
2. Effizienz (Ressourcenverbrauch): der Algorithmus sollte möglichst schnell sein und wenig Speicherplatz verbrauchen.
3. Effektivität: Die Ergebnisse des Algorithmus sollten eine Qualität aufweisen, die den Qualitätsanforderungen bei der Anwendung auf die jeweilige Problemstellung genügt.
4. Robustheit: Die Qualität der Ergebnisse des Algorithmus sollte bei unterschiedlichen Eingaben nicht allzu unterschiedlich ausfallen.

Bei der Beurteilung der Algorithmen hinsichtlich der Eignung zur Anwendung auf die Problemstellung spielt auch die Frage eine Rolle, inwieweit und auf welche Art und Weise der jeweilige Algorithmus in einem interaktiven System angewendet werden kann. Der Algorithmus muss in der Lage sein, nach einer Modifikation des Optimierungsproblems durch den Dispatcher dieses weiter zu optimieren. Hier stellt sich die Frage, welche Teile des Optimierungsproblems nach der Modifikation durch den Dispatcher neu eingelesen und welche Parameter des Algorithmus an die modifizierte Variante des Optimierungsproblems angepasst werden müssen.

Bei Algorithmen, die mit mehreren Lösungen, d. h. einem Pool von Lösungen arbeiten, stellt sich die Frage, wie die Lösungen des Pools mit dem durch den Dispatcher modifizierten Optimierungsproblem zu synchronisieren bzw. an dieses anzupassen sind. Bei einfachen

Modifikationen wie dem Umrouten eines Tourstops fällt die Veränderung nicht stark ins Gewicht. Bei starken Veränderungen wie z. B. der Auflösung einer Tour ohne nachfolgende Synchronisierung des Pools besteht eine hohe Wahrscheinlichkeit, dass der Algorithmus in den Bereich des Lösungsraums zurückspringt, in dem die Lösung des Optimierungsproblems vor der Modifikation liegt. Für den Benutzer würde dies den Eindruck erwecken, dass ihm seine Interaktion verweigert wird. Pools von Lösungen werden meist dazu verwendet, eine gewisse Diversifikation bei der Suche im Lösungsraum zu gewährleisten. Ein Beispiel hierfür ist eine Population von Lösungen bei den genetischen Algorithmen. Eine Anpassung eines Pools von Lösungen an die modifizierte Variante des Optimierungsproblems bedeutet zumindest einen zusätzlichen Aufwand bei der Implementation des Verfahrens.

Sollte der Algorithmus nach Modifikation des Optimierungsproblems dieses nicht weiter optimieren können, muss der Algorithmus mit dem durch den Dispatcher modifizierten Optimierungsproblem neu gestartet werden.

Alle der im Kapitel Grundlagen vorgestellten Algorithmen sind iterative Algorithmen. Für iterative Algorithmen besteht ein Ansatz zur Integration des Algorithmus in ein interaktives System darin, Interaktionen des Benutzers nur nach Ablauf einer Iteration zuzulassen. Dies bedingt jedoch, dass eine einzelne Iteration des jeweiligen Algorithmus schnell abläuft, um die Wartezeit des Dispatchers, der beabsichtigt, eine Aktion durchzuführen, in vernünftigem Rahmen zu halten. Im Bereich des dynamischen Flottenmanagements und der Anwendungsdomäne der Problemstellung sind oft schnelle Entscheidungen gefragt. Eine Wartezeit von mehr als ein bis zwei Sekunden kann hier schon als hinderlich empfunden werden.

Da für die Interaktion des Dispatchers die entsprechenden Modifikationsoperatoren zu implementieren sind, erscheinen diejenigen Algorithmen vorteilhaft, die die Suche im Lösungsraum durch Anwendung der Modifikationsoperatoren durchführen. Der Aufwand für die Implementierung des Programms wird dadurch verringert und das Gesamtsystem insgesamt leichter nachvollziehbar.

Da es bei vielen metaheuristischen Verfahren keine allgemein gültigen Regeln zur Wahl von Werten bei der Parametrisierung des Algorithmus gibt, spielen auch die Erfahrungen des Systemdesigners mit den jeweiligen Metaheuristiken bei der Auswahl der zu verwendenden Metaheuristik eine Rolle. Im folgenden wird auf die Anwendbarkeit der einzelnen im Kapitel Grundlagen vorgestellten Algorithmen eingegangen.

3.3.3.1 Simulated Annealing

Simulated Annealing alleine erscheint aufgrund der schwachen Performanz zunächst als weniger für die Anwendung auf die Problemstellung geeignet. Gute Lösungen sind meist nur mit Annealing–Schedules erreichbar, bei denen die Temperatur sehr langsam abgesenkt

wird, was zu längeren Laufzeiten führt. Bei Annealing–Schedules mit schnellerer Absenkung fängt sich der Algorithmus schnell in einem lokalen Optimum. Allerdings könnte hier eine Intervention des Dispatchers den Algorithmus aus dem lokalen Optimum befreien. Die Modifikationsoperatoren für die Interaktion durch den Dispatcher können auch durch den Algorithmus angewendet werden. Die Laufzeit einer einzelnen Iteration wird bei Simulated Annealing hauptsächlich durch die Bewertungfunktion bestimmt. Kann diese effizient ausgeführt werden, läuft eine einzelne Iteration sehr schnell ab. Nach Interaktion des Dispatchers ist nur die augenblicklich beste Lösung mit der durch den Dispatcher modifizierten Lösung zu synchronisieren. Da der Autor zudem über etwas Erfahrung mit Simulated Annealing verfügt, ist ein Test dieses Algorithmus auf Anwendbarkeit auf die Problemstellung in Erwägung zu ziehen.

3.3.3.2 Tabu–Suche

Die Varianten der Tabu–Suche erzielen bei Vehicle Routing Problemen gute Ergebnisse, allerdings ist der Implementierungsaufwand nicht unerheblich, wenn die Tabu–Liste effizient gestaltet werden soll, z. B. durch Verwendung von Tabu–Tags wie bei LAPORTE U. A. [vgl. Laporte u. a., 2000, S.6]. Der Autor verfügt über keine praktische Erfahrung mit Tabu–Suche. Aus diesen Gründen wird die Anwendung der Tabu–Suche auf die Problemstellung nicht in Erwägung gezogen.

3.3.3.3 Ant Colony Optimization

Der Literatur zufolge erzielen Varianten der Ant Colony Optimization (ACO) Familie bei der Anwendung auf Vehicle Routing Probleme gute Ergebnisse. Allerdings sind bei jeder Iteration umfangreiche Berechnungen notwendig. Die Formel 2.10 zur Berechnung der Auswahlwahrscheinlichkeit einer Kante durch eine Ameise auf Seite 49 veranschaulicht dies. Zudem erscheint eine Integration eines ACO Algorithmus in ein interaktives System als sehr schwierig. Unklar ist hier u. a., mit welcher Pheromonkonzentration neue, durch eine Modifikation des Optimierungsproblems durch den Dispatcher eingeführte Kanten belegt werden. Die Anwendung von ACO Algorithmen auf die Problemstellung erscheint deswegen als problematisch.

3.3.3.4 Large Neighbourhood Search

Mit der Anwendung von Large Neighbourhood Search auf VRP können sehr gute Ergebnisse erzielt werden. Allerdings ist schwer abzuschätzen, wie schnell eine Iteration des Algorithmus abläuft. Zur Entfernung von Tourstops können einfache Operatoren zum Einsatz

kommen. Allerdings werden etliche Tourstops auf einmal entfernt und durch ein Branch and Bound Verfahren wieder eingefügt, was relativ aufwendig ist. Daher erscheint auch die Anwendung des Large Neighbourhood Search auf die Problemstellung als problematisch.

3.3.3.5 Advanced Large Neighbourhood Search

Das Advanced Large Neighbourhood Search (ALNS) gehört zu den State of the Art Algorithmen im VRP–Bereich. Wie beim Large Neighbourhood Search werden pro Iteration etliche Tourstops entfernt und wieder eingefügt. Sowohl für das Entfernen als auch für das Wiedereinfügen werden jeweils unterschiedliche Operatoren benutzt, die mehr oder weniger aufwendig sind. Das Verfahren ist insgesamt sehr aufwendig. Allerdings bezeichnen die Urheber des Verfahrens, PISINGER UND ROPKE, ALNS auch als Framework [vgl. Pisinger und Ropke, 2007, S.3,S.10]. Nicht alle Bestandteile des ALNS müssen zwingend implementiert werden, um einen lauffähigen Algorithmus zu erhalten. Die Implementation der Bestandteile (z. B. Gewichtung der Modifikationsoperatoren, Rauschfunktion) kann stufenweise erfolgen und einzelne Bestandteile können gegeneinander ausgetauscht werden. Zudem kann bei ALNS Simulated Annealing als äußeres Framework benutzt werden. Das Verfahren scheint somit gut zur Anwendung auf die Problemstellung geeignet zu sein.

3.3.3.6 Genetische Algorithmen

Genetische Algorithmen erzielen bei der Optimierung von VRP–Problemen im allgemeinen keine besonders guten Ergebnisse. Ein Hauptproblem ist die Codierung der Lösungen in Chromosomen bei kombinatorischen Problemen. Ein Crossover von Chromosomen führt schnell zu Lösungen, die einige Elemente des Kombinationsproblems doppelt, andere Elemente wiederum gar nicht enthält. Dieses Problem kann nur durch die Anwendung spezieller Crossover–Operatoren [vgl. Ombuki u. a., 2006, S.18–22] oder durch eine dem Crossover folgende Anwendung einer Repair–Function [vgl. u. a. Potvin und Bengio, 1996, S.6–9] verhindert werden. Zudem werden bei den genetischen Algorithmen Pools von Lösungen (Generationen) erzeugt, die im Fall einer Interaktion durch den Dispatcher mit der neuen aktuellen Lösung synchronisiert werden müssen. Insgesamt bedeutet dies einen hohen Aufwand bei der Implementation des Verfahrens mit zugleich wenig Aussicht auf besonders gute Ergebnisse. Die Anwendung von genetischen Algorithmen wird deswegen nicht in Erwägung gezogen.

3.3.3.7 Agentenbasierte Systeme

Agentenbasierte Systeme spielen im Bereich der VRP–Optimierung bisher keine nennenswerte Rolle. Die während der Literaturrecherche gefundenen agentenbasierten Systeme erzielen keine besonders guten Ergebnisse. Der Implementierungsaufwand ist als sehr hoch einzuschätzen. Die Anwendung von agentenbasierten Systemen wird deswegen nicht in Erwägung gezogen.

3.3.3.8 Abschließende Bewertung der Metaheuristiken

Von allen im Kapitel Grundlagen beschriebenen metaheuristischen Verfahren erscheinen das Simulated Annealing und das Advanced Large Neighbourhood Search als die am besten zur Anwendung auf die Problemstellung geeigneten Verfahren. Ausgehend von einer Implementation des Simulated Annealing können Bestandteile des ALNS dem Simulated Annealing hinzugefügt werden, um so einen guten Kompromiss zwischen Geschwindigkeit des Algorithmus und Qualität der Ergebnisse zu erzielen.

3.4 Analyse vorhandener VRP–Benchmarks

Die Formate der im wissenschaftlichen Bereich benutzten VRP–Benchmarks bilden die Randbedingungen der Problemstellung nur teilweise ab. Keiner der dem Autor bekannten Benchmarks ermöglicht die Spezifikation der Dateien, die die Instanzen enthalten, die Beschreibung von Instanzen, die alle Arten von Requests (Pickup–Request, Delivery–Request bzw. Pickup–und–Delivery–Request) enthalten. Die Beschreibung von Instanzen, bei denen bei einem Tourstop mehr als ein Request pro Planungshorizont auftritt, ist meist nicht möglich. Unter realen Bedingungen und im Umfeld der Problemstellung ist es jedoch sehr wohl möglich, dass ein Kunde mehr als einen Request pro Planungshorizont stellt. Die Vergabe von frei wählbaren Constraints (in der Problemstellung die Mindestbesatzung der Transportfahrzeuge) ist ebenfalls bei keinem der bekannten Benchmarks möglich. Dies bedingt die Erstellung eines eigenen Formats zur Beschreibung von Testinstanzen.

Zur Verifikation der Funktionsfähigkeit des Prototypen und zur Einschätzung seiner Leistungsfähigkeit bei der Optimierung von Instanzen der Art der Problemstellung sind Benchmarks aus der wissenschaftlichen Literatur notwendig, da die Instanzen dieser Benchmarks meist von mehreren, teilweise sehr leistungsfähigen Verfahren gelöst werden. Ein Vergleich der dabei erzielten Ergebnisse mit den Ergebnissen aus der Optimierung durch den Prototypen ermöglicht eine Einschätzung der Leistungsfähigkeit des Prototypen. Da kein Benchmark aus dem wissenschaftlichen Bereich alle drei Arten von Requests enthält, bietet es

sich an, den Prototypen sowohl an Instanzen aus Benchmarks zu testen, die Pickup– bzw. Delivery–Requests enthalten, als auch an Instanzen aus Benchmarks zu testen, die Pickup-and–Delivery–Requests enthalten. Dabei sollten die Instanzen jeweils auch Zeitfenster aufweisen. Für den Test geeignet sind u. a. die Instanzen aus dem Solomon Benchmark, dessen Instanzen Pickup– bzw. Delivery–Requests enthalten, sowie der Benchmark von Ropke und Cordeau, dessen Instanzen Pickup–and–Delivery–Requests enthalten.

Um Instanzen der gängigen, im wissenschaftlichen Bereich benutzten Benchmarks und von Instanzen VRP aus dem Bereich der Problemstellung beschreiben zu können, muss das Format folgende Möglichkeiten bieten:

- Beschreibung der Koordinaten, des Zeitfensters, der Aufnahmekapazität der Schleusen des Depots (in Anzahl an Fahrzeugen) sowie der zur Abfertigung von Fahrzeugen nötigen Bearbeitungszeit bei der Abfahrt vom Depot sowie der Anfahrt zum Depot.
- Beschreibung der Koordinaten der Kunden.
- Beschreibung von Requests mit Angabe von Request–Art (Pickup–Request, Delivery–Request sowie PD–Request), Constraints (in der Problemstellung die nötige Min- destanzahl an Besatzungsmitglieder des Transporters zur Abwicklung des Requests), Menge des zu transportierenden Wertes, des Kunden, sowie Zeitfenster des Kunden und die zur Abwicklung beim Kunden nötigen Zeitspanne. Bei PD–Requests sind beide Kunden anzugeben.
- Anzahl und Kapazität der Fahrzeuge sowie die Kosten pro Entfernungs– und Zeiteinheit für ein Fahrzeug.

Zur Beschreibung von Daten und zum Datenaustausch zwischen Computer–Systemen hat sich XML als Standard etabliert. Durch die Beschreibung der Daten durch die Daten einschließende spezielle Markup–Strings (den sogenannten *Tags*) werden XML–Dateien schnell groß und unübersichtlich. Dies ist insbesondere dann der Fall, wenn viele Werte in jeweils einem Tag anzugeben sind. Das Erstellen von Testintanzen im XML–Format von Hand wird selbst bei Instanzen mit wenigen Tourstops und selbst bei Verwendung eines XML–Editors schnell mühselig. Allerdings bietet XML den Vorteil, Daten anhand eines *Schema*, das die Spezifikation des XML–Formats beschreibt, zu validieren. Dadurch können Instanz–Dateien sowohl auf die Einhaltung der Spezifikation des zu erstellenden VRP–Formats als auch auf Gültigkeit der Daten (z. B. Datentyp, Wertebereich) überprüft werden.

In dieser Situation erscheint folgende Vorgehensweise sinnvoll: die für die Problemstellung benötigte Spezifikation der Instanz–Dateien wird sowohl im Plaintext–Format als auch im XML–Format erstellt. Das Plaintext–Format erlaubt die leichte Erstellung von kleineren Testinstanzen mit einem einfachen Texteditor. Für größere Instanzen muss ein Generator realisiert werden, der Testinstanzen anhand von Verteilungen von Kunden und Requests sto-

chastisch generiert. Für Instanzdateien im Plaintext–Format der Spezifikation und für Benchmarks aus der wissenschaftlichen Literatur sind Parser zu realisieren, die das jeweilige Format der Instanzdatei in das XML–Format der Spezifikation umwandeln. Als Eingabe für den Prototypen werden nur Instanzdateien im XML–Format der Spezifikation verwendet.

4 Design

In diesem Kapitel erfolgt auf Basis der in Kapitel 3 erfolgten Analyse das Design des Systems. Ausgehend von der fachlichen Architektur, die den allgemeinen Aufbau des Systems und die Aufteilung des Systems in seine Hauptbestandteile beschreibt, wird die technische Architektur entwickelt, in der auf den Aufbau der einzelnen Hauptbestandteile näher eingegangen wird. Da die Realisierung des Prototypen auf Grundlage des Frameworks SAF (*Stochastics Algorithm Framework*), das während der Studienarbeit des Autors entstand, stattfindet, wird dabei auch auf die Architektur von SAF eingegangen.

4.1 Fachliche Architektur

4.1.1 Entwurfsstudie eines Systems zur Optimierung dynamisch stochastischer VRP

Abbildung 4.1 zeigt eine Entwurfsstudie eines Systems zur Optimierung dynamisch stochastischer VRP. Das System umfasst die folgenden Komponenten: die Datenstruktur des enthält die Daten der a–priori und spontanen Requests, die geographischen Positionen der Kunden, der aktuellen geographischen Position und Status der Transportfahrzeuge sowie einen Plan, der die aktuellen Touren enthält. Über ein graphisches User Interface können Request–Daten ausgewählt bzw. eingegeben werden. Das User Interface zeigt das zu Optimierende DSVRP an und bietet die Funktionalität zur Interaktion bei der Optimierung an. Der DSVRP Solver besteht aus einem Algorithmus zur Optimierung des DSVRP und gegenenfalls zusätzlich aus einer Konstruktionsheuristik zur Erzeugung von Startlösungen. Das User Interface bietet die Einstellung von Parametern des Algorithmus wie z. B. Parameter zur Vergabe von Time Slacks in Touren an.

Eine Knowledge Database stellt Kundendaten wie Standorte der Kunden, Requests aus der Vergangenheit sowie Daten über den Ablauf vergangener Planungshorizonte bereit. Zur Ermittlung von Distanzen zwischen geographischen Punkten dient ein geographisches Informationssystem (GIS). Die Abschätzung der Reisedauer zwischen zwei Punkten erfolgt durch das Subsystem Travel Time Prognosis. Dieses kann sowohl auf den Datenbestand der Knowledge Database als auch auf das geographische Informationssytem zugreifen.

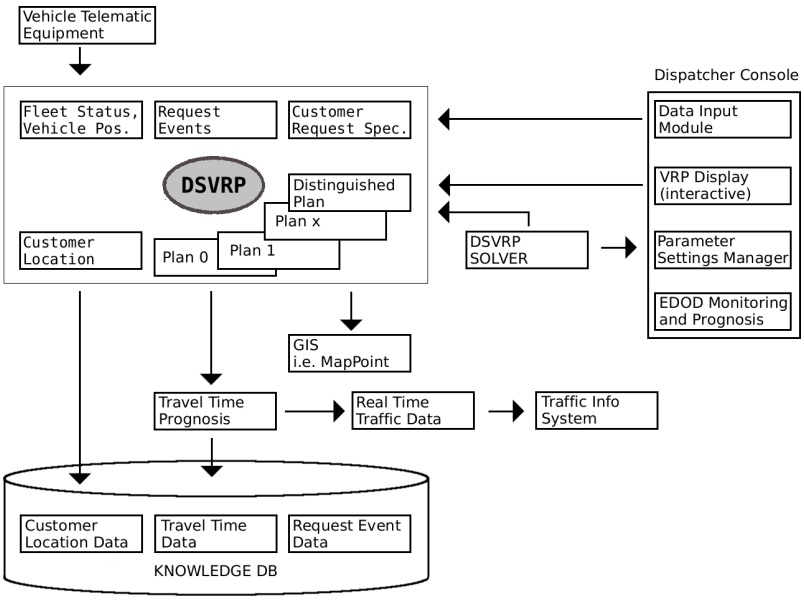

Abbildung 4.1: Entwurfsstudie eines Systems zur Optimierung dynamisch stochastischer VRP

Telematisches Equipment in den Transportfahrzeugen dient zur Übermittlung der aktuellen Position der Transportfahrzeuge und zur Kommunikation mit den Transportfahrzeugen, basierend auf einem geographischen Positionssystem wie z. B. GPS (*Global Positioning System*) sowie Kommunikationsequipment wie z. B. GPRS (*General Packet Radio System*). Ein Subsystem für das EDOD Monitoring überwacht die bisherige Entwicklung der Dynamik während des aktuellen Planungshorizonts und prognostiziert anhand der in der Knowledge Database gesammelten Daten über die Abläufe vergangener Planungshorizonte deren weiteren Verlauf.

4.1.2 Entwurfstudie des Prototypen

Eine Realisierung der gesamten Funktionalität des in Abbildung 4.1 dargestellten Systems ist im Rahmen dieser Arbeit nicht möglich. Abbildung 4.2 zeigt die im Prototypen verwirklichte Funktionalität der ersten Entwurfstudie. Das geographische Informationssystem und das Subsystem zur Abschätzung der Reisedauer zwischen zwei Punkten sind durch Testtreiber ersetzt. Der Testtreiber für das geographische Informationssystem gibt für zwei Punkte mit Koordinaten die Distanz im zweidimensionalen euklidischen Raum zurück. Der Testtreiber zur Abschätzung der Reisedauer gibt für zwei Punkte die Distanz, multipliziert mit einem konfigurierbaren Adaptionsfaktor, zurück.

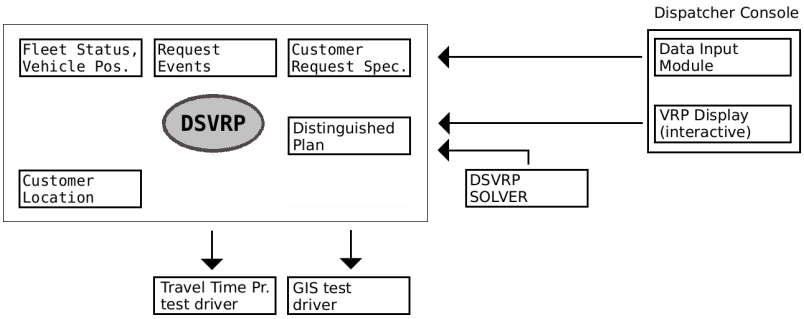

Abbildung 4.2: Abgeleitete Entwurfsstudie für das zu erstellende Systems

4.1.3 Entwicklung des Systems mit dem Framework SAF

Der Prototyp des zu erstellenden Systems wird auf Grundlage des Frameworks *SAF* (Stochastic Algorithms Framework) für stochastische Algorithmen, das während der Studienarbeit des Autors entstand und unter der GNU Public License veröffentlicht ist, erstellt [vgl. Held, 2008]. SAF selbst beinhaltet bisher keine Funktionalität zur interaktiven Lösung von Optimierungsproblemen, das Design des Frameworks lässt jedoch die Integration dieser Funktionalität zu. Das Fachliche Design orientiert sich daher am Design des SAF. Abbildung 4.3 zeigt die Architektur des SAF.

Die Hauptkomponenten des Systems SAF bestehen aus den Klassen *OptimizationProblem*, *OptimizationAlgorithm*, *InitialSolutionProvider* (Generator für Startlösungen) sowie der Klasse *RandomSource* (Zufallszahlen–Quelle). Der jeweils eingesetzte Optimierungsalgorithmus kann durch eine Ablaufsteuerung (*AlgorithmRunControl* im Bild) vorübergehend angehalten und im Einzelschritt–Modus, bei dem bei iterativen Algorithmen jeweils eine Iteration pro Anforderung des Benutzer durchgeführt wird, betrieben werden. Zwei Displays (*DataDisplay* und *GraphicDisplay* im Bild) dienen zur numerischen und graphischen Anzeige des Optimierungsproblems. Das numerische Display kann dabei auch Daten des Zustands anderer Komponenten, wie z. B. Parametereinstellungen des Optimierungsalgorithmus, anzeigen. Konfigurationen, die aus einer Zusammenstellung von Instanz des Optimierungsproblems, Optimierungsalgorithmus, Zufallszahlenquelle und gegebenfalls Generator für Startlösungen bestehen, können über das graphische Benutzerinterface oder durch Auswahl einer Konfigurationsdateien eingegeben werden. Das Einlesen von Konfigurationsdateien geschieht über die systeminternen Parser (*Configuration File Parser* im Bild). Für das Einlesen von Dateien, die Instanzen von Optimierungsproblemen definieren, bestehen Klassen von Parsern, von denen neu zu implementierende Parser abgeleitet werden können (*Problem File Parser* im Bild).

Das System bietet die Funktionalität, Werte von Parametern des Algorithmus und des Optimierungsproblems während des Laufs des Algorithmus in Log–Dateien zu schreiben (*Log-*

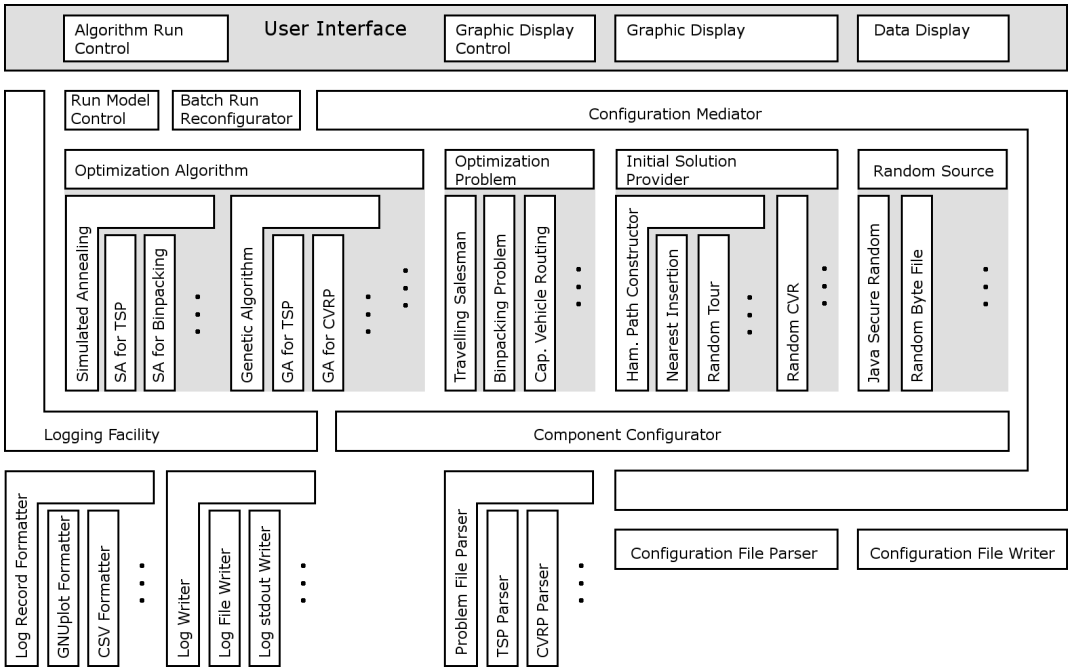

Abbildung 4.3: Architektur des Frameworks für stochastische Algorithmen SAF. Graphik aus [Held, 2008, S.8]

ging Facility im Bild). Die Konfiguration von Batch Runs, bei denen mehrere Abläufe eines Optimierungsalgorithmus vorkonfiguriert automatisch ablaufen, ist ebenfalls möglich (*Batch Model Control* und *Batch Run Reconfigurator* im Bild).

Um die geforderte Funktionalität zur interaktiven Optimierung in das System integrieren zu können, wurde eine neue Komponente eingeführt. Über den *User Action Router*, in Abbildung 4.4 durch einen Pfeil gekennzeichnet, kann der Benutzer interaktiv den Zustand des Optimierungsproblems verändern. Die Architektur folgt hier dem Facade–Pattern von GAMMA U. A.. Die Einführung des User Action Router als einziger Zugangspunkt zum Optimierungsproblem für die interaktive Veränderung des Zustands des Optimierungsproblems bringt den Vorteil, dass sämtliche Zugriffe auf das Optimierungsproblem zur interaktiven Veränderung zentral gesperrt werden können.

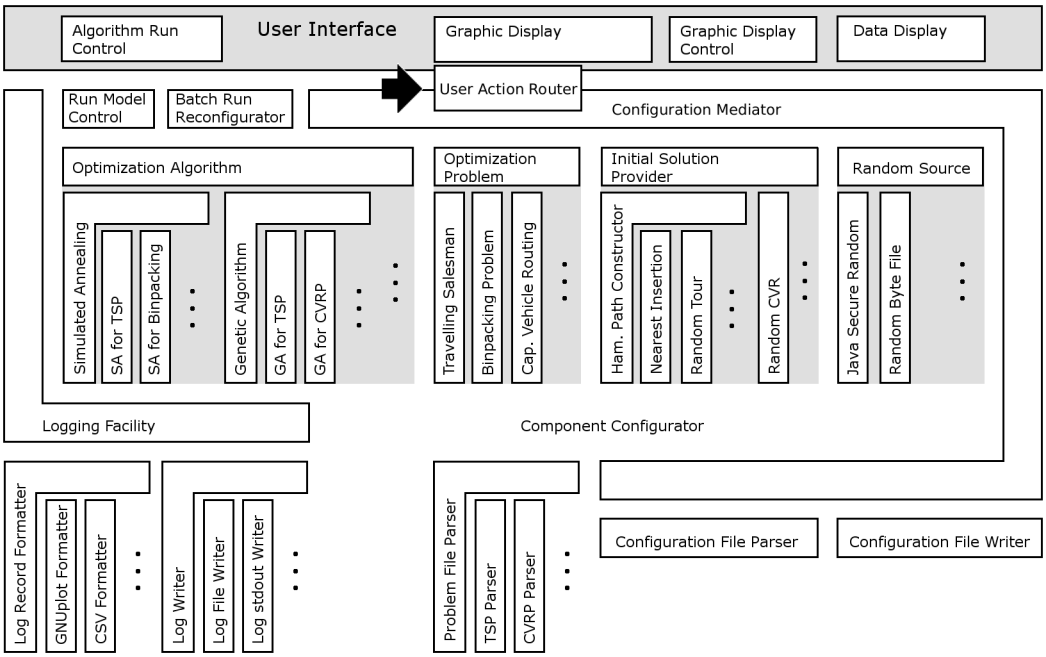

Abbildung 4.4: Integration des UserActionRouter in das Framework SAF

4.2 Technische Architektur

4.2.1 Modellierung des Optimierungsproblems

In diesem Abschnitt wird der Aufbau der einzelnen Komponenten des Systems, ausgehend von den Komponenten, die das Optimierungsproblem darstellen, erläutert. Abbildung 4.5 zeigt das Klassenmodell der Komponenten des Optimierungsproblems, das durch die Klasse *DSVRP* repräsentiert wird. Ein *Plan* ist eine Lösung des Optimierungsproblems und enthält Touren, denen jeweils ein Fahrzeug (Klasse *Vehicle* im Bild) zugeordnet ist. Die weiteren Komponenten werden im Folgenden erläutert.

4.2.1.1 Modellierung der Touren

Die Modellierung von Touren ist ein besonders kritischer Punkt bei der Modellierung des Systems, da bei der Optimierung des Systems hauptsächlich auf Touren zugegriffen und diese verändert werden. Dies bedingt, dass das Modell der Touren möglichst einfach gestaltet ist. Andererseits sollte das Modell der Touren flexibel genug sein, um Randbedingung wie z. B. die Bearbeitungszeiten der Fahrzeuge in den Schleusen des Depots abbilden zu können.

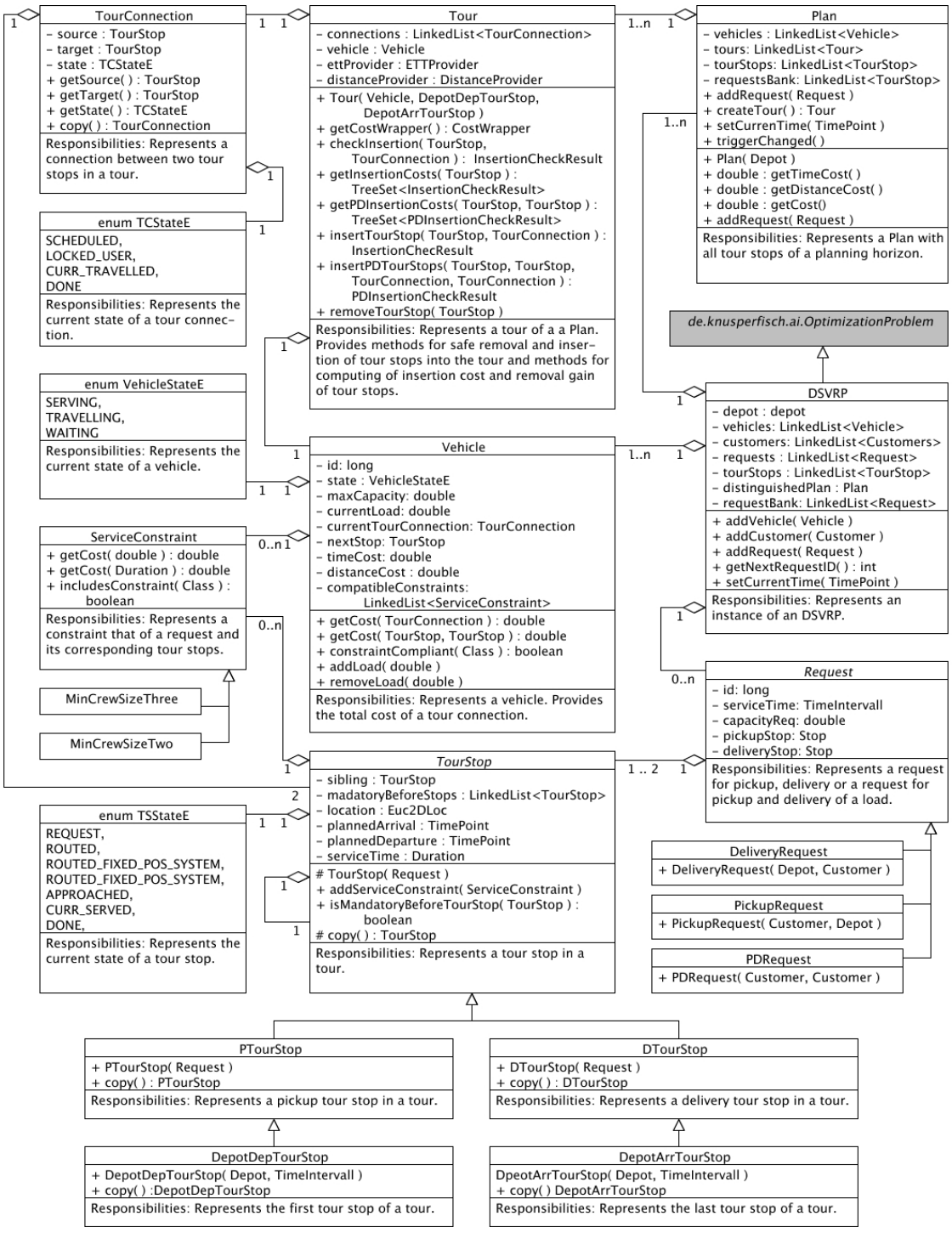

Abbildung 4.5: Klassenmodell der Komponenten des Optimierungsproblems

Um mehrere Requests eines Kunden während eines Planungshorizonts zu ermöglichen, ist es sinnvoll, nicht Kunden in eine Tour aufzunehmen, sondern Objekte, die eine Verbindung zwischen Kunde und Request darstellen. Dies sind die in Kapitel 3, (Analyse) eingeführten Tourstops. Bei den Tourstops wird zwischen Pickup–TourStops (Klasse *PTourStop* im Bild) und Delivery–TourStops (Klasse *DTourStop* im Bild) unterschieden. Erstere sind Tourstops, bei denen Werte vom Transportfahrzeug aufzunehmen sind, letztere sind Tourstops, bei denen Werte auszuliefern sind.

Ein erster Ansatz besteht darin, Touren als eine geordnete Folge von Tourstops zu modellieren. Bei der Modifikation von Touren werden Tourstops aus einer Tour entfernt und in die gleiche Tour an anderer Stelle bzw. in eine andere Tour wieder eingefügt. Anstatt die Modifikation beim Einfügen eines Tourstops so aufzufassen, dass ein Tourstop vor bzw. nach einem anderen Tourstop eingefügt wird, besteht auch die Möglichkeit, das Einfügen eines Tourstops grundsätzlich als Einfügung in eine Tourverbindung aufzufassen. Tourverbindungen werden im System durch die Klasse *TourConnection* repräsentiert. Ein Objekt der Klasse Tour Connection hält jeweils eine Referenz zum Ursprungs–Tourstop (im folgenden *Source* genannt) sowie zum Ziel–Tourstop (im folgenden *Destination* genannt), die jeweils Anfang und Ende der Tourverbindung darstellen.

Bei der Modifikation einer Tour sind umfangreiche Berechnungen zur Überprüfung der Einhaltung der Randbedingungen wie z. B. Zeitfenster oder Kapazitätsbeschränkungen nötig. Die Programmierung der dazu benötigten Methoden wird durch die Einführung von Tour Connections, die jeweils eine Verbindung zwischen zwei Tourstops darstellen, vereinfacht und leichter nachvollziehbar.

Die graphische Anzeige des Optimierungsproblems ist nicht zuletzt wegen der geforderten Interaktionsmöglichkeit des Dispatcher ein wichtiger Aspekt bei der Modellierung der Tour. Durch die Einführung von Tour Connections wird sowohl die Anzeige des Zustands einer Tourverbindung als auch die Realisierung der Interaktionsmöglichkeit zur Modifikation einer Tourverbindung erleichtert und der Aufbau dieses Bestandteils des Systems ebenfalls leichter nachvollziehbar.

4.2.1.2 Modellierung der Constraints für Kundenklassen

In der Problemstellung lassen sich die Stops und die sich jeweils daraus ergebenden TourStops in jeweils zwei Klassen unterteilen: eine Klasse bei der zur Bearbeitung eines Requests die Mindestbesatzungsstärke des Transportfahrzeugs zwei Mann betragen muss, sowie eine Klasse bei der zur Beaabeitung eines Requests eine Mindestbesatzungsstärke von drei Mann nötig ist. Von der Klasse *Service Constraint* sind zwei Klassen, und *MinCrewSizeThree*, abgeleitet, die die entsprechenden Constraints repräsentieren. Diese Constraints

werden an Tourstops und Fahrzeuge vergeben. Ein Fahrzeug kann diejenigen Tourstops bedienen, für die es die entsprechenden Constraints enthält. Der Constraint MincrewSizeThree enthält den Constraint MinCrewSizeTwo, d. h. ein Fahrzeug, das den Constraint MinCrewSizeThree enthält, kann auch Tourstops mit dem Constraint MinCrewSizeTwo bedienen. Beim Versuch, einen Tourstop in eine Tour einzufügen, wird anhand der Constraints, die ein Fahrzeug enthält, geprüft, ob der Tourstop in die Tour eingefügt werden kann. Bei einer Constraintverletzung verweigert das System das Einfügen des Tourstops in die Tour.

4.2.1.3 Zustände von Tourstops und Tour Connections

Während eines Planungshorizonts durchlaufen Tourstops und Tour Connnections verschiedene Zustände, durch die u. a. bestimmt ist, ob der jeweilige Tourstop bzw. die jeweilige Tour Connection aus einer Tour entfernt wird bzw. ob in die jeweilige Tour Connection ein Tourstop eingefügt werden kann. So muss z. B. bei einem schon bearbeiteten Request verhindert werden, dass der dem Request zugeordnete Tourstop aus der Tour durch den Optimierungsalgorithmus oder durch Interaktion des Dispatchers innerhalb der Tour verschoben oder aus der Tour entfernt wird. Fährt ein Fahrzeug einen Tourstop an, darf in die Tour Connection, die das Fahrzeug gerade befährt, kein Tourstop eingefügt werden. Das Benutzerinterface für den Dispatcher zeigt die graphischen Elemente zur Repräsentation der Tourstops und Tour Connections durch verschiedene Farben an. Abbildung 4.6 zeigt die Zustände eines Tourstops während eines Planungshorizonts.

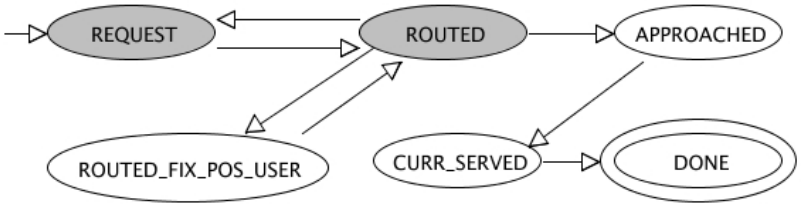

Abbildung 4.6: Zustände eines Tourstop während des Planungshorizonts

Die einzelnen Zustände sind dabei:

- **REQUEST**: Der Tourstop ist in keiner Tour enthalten.
- **ROUTED**: Der Tourstop ist in einer Tour enthalten. Dies ist der einzige Zustand, in dem der Tourstop innerhalb der Tour verschoben oder aus der Tour entfernt werden kann.
- **ROUTED_FIX_POS_USER**: Der Tourstop ist in einer Tour enthalten und wurde vom Dispatcher für Modifikationen der Tour gesperrt, d. h. der Tourstop darf nicht innerhalb der Tour verschoben oder aus der Tour entfernt werden.

- **APPROACHED**: Der Tourstop wird gerade von einem Transportfahrzeug angefahren, d. h. das Transportfahrzeug befindet sich auf direktem Weg zum Tourstop.
- **CURR_SERVED**: Der Tourstop wird gerade von einem Transportfahrzeug bearbeitet, d. h. Werte werden gerade ein– bzw. ausgeladen.
- **DONE**: Die Bearbeitung des Tourstops ist abgeschlossen. Das Transportfahrzeug hat den Tourstop verlassen und befindet sich auf dem Weg zum nächsten Tourstop.

Abbildung 4.7 zeigt die Zustände einer Tour Connection während eines Planungshorizonts.

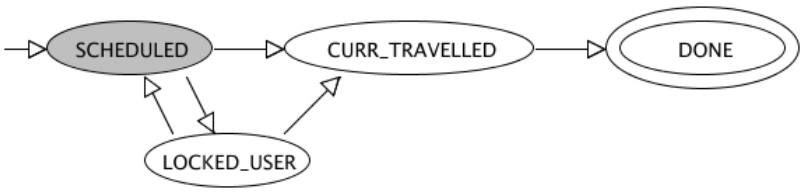

Abbildung 4.7: Zustände einer Tour Connection während des Planungshorizonts

Die einzelnen Zustände sind dabei:

- **SCHEDULED**: Dies ist der einzige Zustand, in dem die Tour Connection entfernt werden kann bzw. in dem ein Tourstop in die Tour Connection eingefügt werden kann.
- **LOCKED_USER**: Die Tour Connection ist vom Dispatcher für Modifikationen (Entfernung der Tour Connection bzw. Einfügen eines Tourstops in die Tour Connection) gesperrt.
- **CURR_TRAVELLED**: Die Tour Connection wird gerade von einem Transportfahrzeug befahren.
- **DONE**: Das Fahrzeug, das die Tour Connection bisher befahren hat, hat die Tour Connection verlassen und ist beim Ziel–Tourstop der Tour Connection (Destination) angekommen.

4.2.2 Modellierung des Depots

Die Modellierung des Depots stellt aufgrund der Randbedingung der Bearbeitungszeiten der Fahrzeuge in der Schleuse bei der Abfahrt vom und der Anfahrt zum Depot eine besondere Herausforderung dar. Den Fahrzeugen sollten möglichst diejenigen Bearbeitungszeiten in der Schleuse zugewiesen werden, bei denen die Wartezeit für die Fahrzeuge beim ersten

Kunden bis zur Bearbeitung des Kunden sowie die Wartezeit bei der Rückkehr zum Depot bis zur Bearbeitung in der Schleuse möglichst minimal sind. Die Bearbeitungszeiten in der Schleuse werden nachfolgend mit *Schleusenzeiten* bezeichnet. Für die Problemstellung wird für alle Touren eine Schleusenzeit von 20 Minuten beim Verlassen des Depots sowie 40 Minuten bei der Anfahrt zum Depot angenommen. Die Schleusen des Depots weisen eine Kapazität in Fahrzeugen auf, d. h. durch die Kapazität der Schleuse ist die Anzahl an Fahrzeugen definiert, die sich gleichzeitig in der Schleuse befinden können.

Abbildung 4.8 zeigt ein Depot mit einer Schleuse, in der drei Fahrzeuge gleichzeitig bearbeitet werden können und einer Warteschlange, in der Fahrzeuge auf die Bearbeitung in der Schleuse warten.

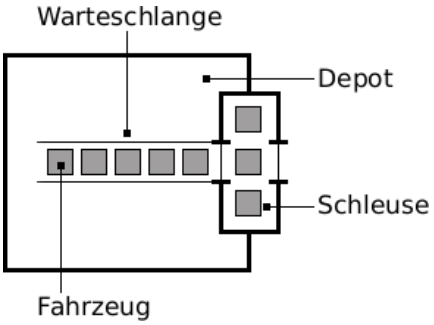

Abbildung 4.8: Depot mit einer Schleuse und einer Warteschlange

Zur leichteren Berechnung der für ein Fahrzeug günstigsten möglichen Schleusenzeiten ist es vorteilhaft, die Schleuse des Depots in eine virtuelle Schleuse mit virtuellen Warteschlangen aufzuteilen. Dabei wird eine Schleuse in so viele virtuelle Schleusen zerlegt, wie die Kapazität der (realen) Schleuse in Fahrzeugen beträgt. Abbildung 4.9 zeigt das gleiche Depot wie jenes in Abbildung 4.8, mit Aufteilung der Schleuse in drei virtuelle Schleusen und drei Warteschlangen.

Jeder virtuellen Schleuse wird eine eigene Zeitachse zugeteilt, aus der Intervalle für die Schleusenzeiten vergeben werden können. Beim Erzeugen einer neuen Tour werden vor Start der Konstruktionsheuristik dem Depot Departure Tourstop die füheste mögliche Schleusenzeit, dem Depot Arrival Tourstop die späteste mögliche Schleusenzeit zugewiesen. Abbildung 4.10 veranschaulicht den Vorgang. Zur einfacheren Darstellung wird ein kurzer Planungshorizont mit vier Stunden von acht bis zwölf Uhr gewählt.

Nachdem eine Tour durch Konstruktionsheuristik erstellt ist, stehen die Zeiten für den spätesten möglichen Abfahrtszeitpunkt beim Verlassen des Depots sowie der frühesten möglichen Ankunftszeit bei der Anfahrt zum Depot fest. Der späteste mögliche Abfahrtszeitpunkt ist durch den spätesten Zeitpunkt des Zeitfensters des ersten Tourstops in der Tour sowie durch die Fahrtdauer vom Depot zum ersten Tourstop bestimmt. Der früheste mögliche Ankunftszeitpunkt bei der Ankunft zum Depot ist durch den Zeitpunkt, an dem die Bearbeitung

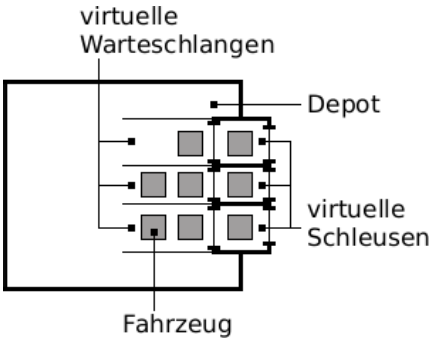

Abbildung 4.9: Aufteilung von Schleuse und Warteschlange des Depots in virtuelle Schleusen und virtuelle Warteschlangen

Abbildung 4.10: Zuteilung der Schleusenzeit bei Erzeugung einer neuen Tour

des letzten Tourstops abgeschlossen ist, sowie der Fahrtdauer vom letzten Tourstop zum Depot bestimmt. Die Schleusenzeiten für die Bearbeitung des Fahrzeugs beim Verlassen des Depots und bei der Ankunft am Depot werden daraufhin, soweit möglich, an die späteste Abfahrtszeit und früheste Ankunftszeit angepasst. Die Abbildungen 4.11 und 4.12 veranschaulichen den Vorgang.

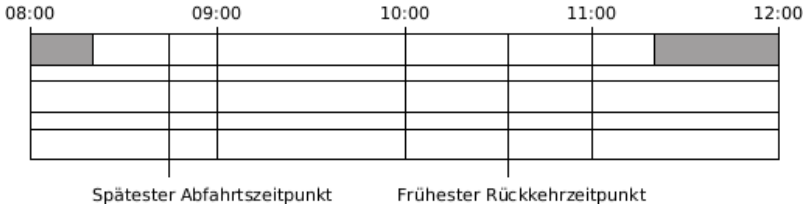

Abbildung 4.11: Spätester Abfahrtszeitpunkt und frühester Ankunftszeitpunkt nach Erstellung der Tour

Die Zeitachsen der virtuellen Schleusen werden bei der Vergabe von Schleusenzeiten in Zeitabschnitte geteilt, so dass bei der Vergabe von Schleusenzeiten für das Verlassen des Depots entweder zwischen den Schleusenzeiten freie Zeitintervalle von der Länge der Bearbeitungsdauer beim Verlassen des Depots bestehen bleiben, oder aber die Schleusenzeiten

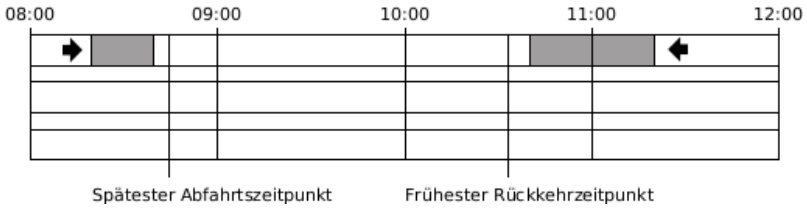

Abbildung 4.12: Zuteilung der Schleusenzeiten anhand des spätesten Abfahrtszeitpunktes und des frühesten Ankunftszeitpunktes

an einander anschließen. Bei der Vergabe von Schleusenzeiten für die Rückkehr zum Depot wird ebenfalls dem entsprechend verfahren. Diese Zeitabschnitte werden im Folgenden *Timeslots* genannt. Werden diese Timeslots bei der Vergabe von Schleusenzeiten nicht berücksichtigt, können freie Zeitintervalle entstehen, die zu kurz für die Bearbeitung der Fahrzeuge beim Verlassen des Depots bzw. der Ankunft am Depot sind. Abbildung 4.13 zeigt die Vergabe von Schleusenzeiten bei Erzeugung einer neuen Tour ohne Berücksichtigung von Timeslots. Dabei entstehen Zeitintervalle, in denen die virtuelle Schleuse des Depots nicht genutzt werden kann. Zudem wird im Verlauf der Erzeugung mehrerer Touren mit zunehmender Anzahl der Touren die maximal mögliche Tourlänge schneller kürzer als bei der Vergabe der Schleusenzeit unter Berücksichtigung von Timeslots. Die Vergabe von Schleusenzeiten für die gleichen Touren wie in Abbildung 4.13, jedoch unter Berücksichtigung von Timeslots, ist in Abbildung 4.14 dargestellt. Der Nachteil bei der Vergabe von Schleusenzeiten unter Berücksichtigung von Timeslots ist die Möglichkeit von längeren Wartezeiten beim ersten Tourstop bzw. bei der Ankunft am Depot nach Abarbeitung der Tour. Wartezeiten bei der Ankunft zum Depot können jedoch als Time Slacks genutzt werden. Zudem besteht die Möglichkeit, vor Beginn des Planungshorizonts die Schleusenzeiten ohne Berücksichtigung der Timeslots optimal anzupassen. Diese Möglichkeit ist im Prototypen noch nicht implementiert.

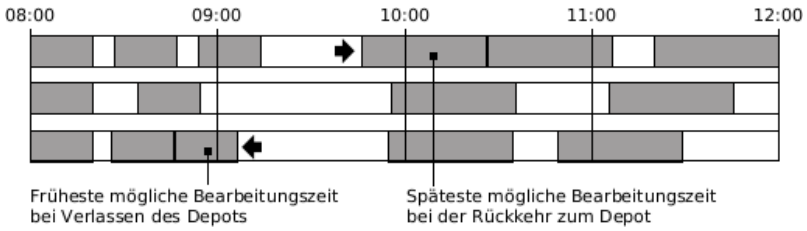

Abbildung 4.13: Zuteilung der Schleusenzeiten ohne Berücksichtigung von Timeslots

Abbildung 4.15 zeigt das Klassenmodell des Depots. Das *Depot* stellt Methoden für Anfragen zur Vergabe von Schleusenzeiten zur Verfügung. Die Depotschleuse wird durch die abstrakte Klasse *Depotmodel* repräsentiert. In den konkreten Unterklassen befindet sich die Implementierung der Funktionalität zur Verwaltung und Vergabe von Schleusenzeiten.

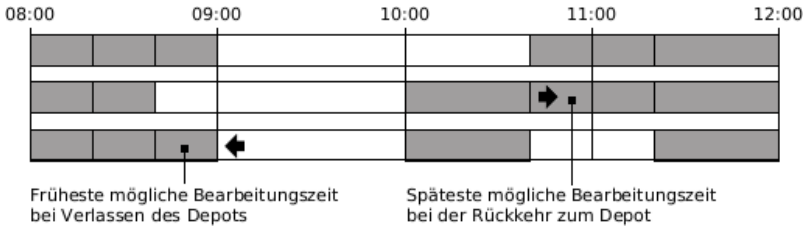

Abbildung 4.14: Zuteilung der Schleusenzeiten unter Berücksichtigung von Timeslots

Das *SafetyLockDepotModel* implementiert die Funktionalität zur Verwaltung und Vergabe von Schleusenzeiten anhand der Randbedingungen der Problemstellung. Das *SimpleDepotModel* ist ein Dummy–Modell, das alle Anfragen zur Vergabe von Schleusenzeiten erfüllt. Der Rückgabewert ist dabei der Zeitpunkt zum Verlassen des Depot bzw. der Rückkehr zum Depot, der bei der Anfrage übergeben wurde.

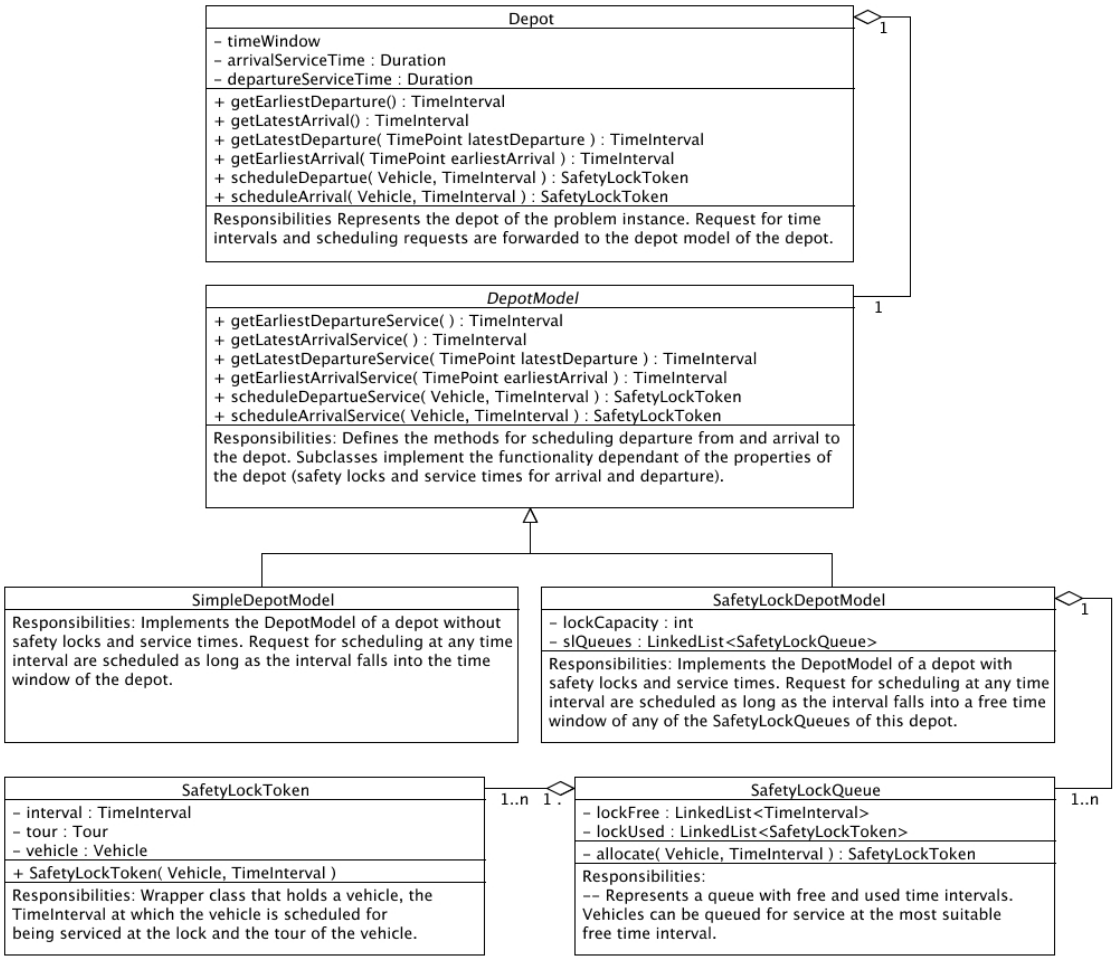

Abbildung 4.15: Klassenmodell des Depots

4.2.3 Erzeugung neuer Touren

4.2.4 Ermittlung der räumlichen und zeitlichen Distanzen

Zur Ermittlung der räumlichen und zeitlichen Distanzen (zu erwartendende Fahrtzeiten) zwischen zwei Tourstops werden in Tourenplanungssystemen meist geographische Informationssysteme verwendet. Zur Ermittlung von zeitlichen Distanzen können, soweit verfügbar, Echtzeit–Daten über das Verkehrsaufkommen und Daten über das durchschnittliche Verkehrsaufkommen zu bestimmten Tageszeiten verwendet werden. Die Einbindung solcher Systeme in den Prototypen kann in dieser Arbeit nicht verwirklicht werden. Zur Ermittlung der räumlichen und zeitlichen Distanzen werden deswegen Testtreiber verwendet, die die entsprechenden Funktionen eines geographischen Informationssystems sowie eines Systems zur Prognose der zeitlichen Distanz vereinfacht realisieren. Die Testtreiber des Prototypen liefern als räumliche und zeitliche Distanz zwischen zwei Tourstops die euklidische Distanz zwischen den beiden Tourstops. Beide Testtreiber können so konfiguriert werden, dass die räumliche bzw. zeitliche Distanz vor der Rückgabe an die anfragende Komponente mit einem Adaptionsfaktor multipliziert wird. Dadurch ist eine Anpassung der Testtreiber an die verschiedenen Benchmarks möglich. Abbildung 4.16 zeigt das Klassenmodell der Testtreiber. Die abstrakte Klasse *DistanceProvider* liefert die räumliche Distanz zwischen zwei Tourstops, die abstrakte Klasse *ETTProvider* (*Exspected Travel Time Provider*) liefert die zeitliche Distanz zwischen zwei Tourstops. Die Klassen *Euc2DDistanceProvider* und *Euc2DETTProvider* sind die konkreten Implementationen der abstrakten Oberklassen.

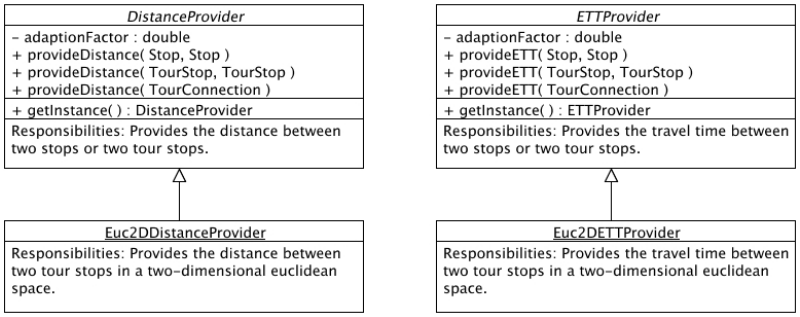

Abbildung 4.16: Klassen der Testtreiber zur Ermittlung der räumlichen und zeitlichen Distanzen

5 Realisierung

In diesem Kapitel wird auf die Implementierung des Systems anhand der in Kapitel 3 (Analyse) entwickelten Anforderungen und des in Kapitel 4 (Design) entwickelten Designs eingegangen. Von der Beschreibung des MRVRP–Formats und der Parser für Instanzdateien ausgehend, werden die für die Simulation benötigten Komponenten vorgestellt. Anschließend werden das Benutzer–Interface und die Interaktionsmöglichkeiten für den Dispatcher dargestellt. Das System wird mit Instanzen, die Anwendungsfällen der Problemstellung entsprechen, getestet. Zum Schluss erfolgt eine Bewertung der bei den Tests erzielten Ergebnisse.

5.1 Zeitmodell des Systems

Bei der Optimierung von VRP aus der Problemstellung sind zur Bewertung einer Lösung, zur Überprüfung einer Lösung auf Gültigkeit sowie zur Entscheidung, ob eine Modifikationsoperation eine ungültige Lösung erzeugt, umfangreiche Berechnungen nötig, bei denen zeitliche Aspekte eine wesentliche Rolle spielen. Die Programmiersprache Java stellt zur Repräsentation von Zeitpunkten die Klasse *GregorianCalendar* zur Verfügung [1]. Die Handhabung der Klasse, wie z. B. das Erstellen von Objekten, die einen Zeitpunkt repräsentieren, ist jedoch sehr umständlich und die Funktionalität der Klasse für die Problemstellung nicht ausreichend. Deswegen wurde von der Klasse GregorianCalendar die Klasse *TimePoint* abgeleitet und mit Methoden versehen, durch die die Programmierung von Berechnungen, bei denen zeitliche Aspekte mit berücksichtigt werden, einfacher gestaltet werden kann.

Zeitpunkte werden in GregorianCalendar und TimePoint durch die Anzahl der Millisekunden, die seit dem 1.1.1970, 0:00 Uhr (auch *Unix-Epoche* genannt) verstrichen sind, repräsentiert. Dadurch ist auch die Zeiteinheit, mit der das System arbeitet, definiert und beträgt eine Millisekunde. Neben der Klasse TimePoint wurden zwei weitere Klassen implementiert, mit denen zeitliche Aspekte der Problemstellung im System abgebildet werden können. Die Klasse *TimeInterval* repräsentiert einen Zeitraum zwischen einem genau definierten Anfangszeitpunkt und Endzeitpunkt, wobei das Intervall Anfangs– und Endzeitpunkt miteinschliesst.

[1] Die früher oft in Java–Programmen benutzte Klasse *java.util.Date* ist mittlerweile als *deprecated*, d. h. veraltet, gekennzeichnet

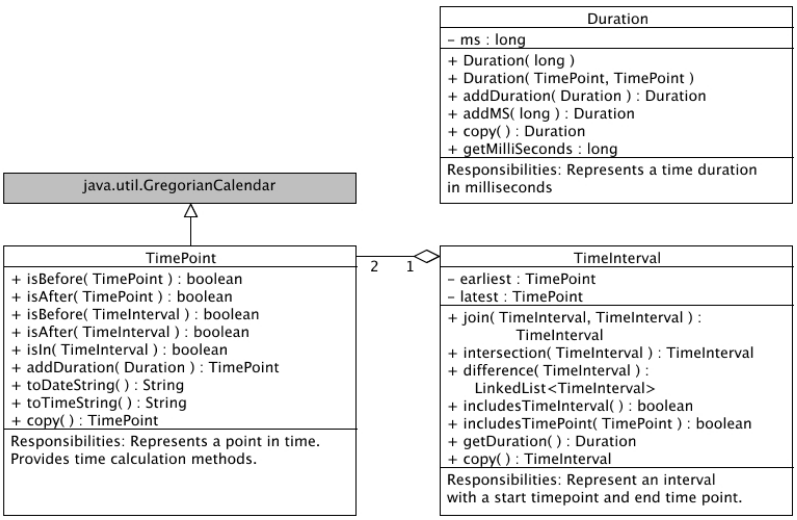

Abbildung 5.1: Zeitmodel des Systems

Diese Klasse wird u. a. für die Repräsentation von Zeitfenstern von Tourstops benutzt. Die Klasse *Duration* repräsentiert eine zeitliche Dauer, bei der Anfangs– und Endzeitpunkt nicht definiert sind. Diese Klasse wird z. B. dazu benutzt, um die geschätzte Fahrtdauer zwischen zwei Tourstops zu repräsentieren.

5.1.1 Simulation des Planungshorizonts

Die Simulation des Planungshorizonts ist auf einfache Weise realisiert. Für die Tests wird angenommen, dass ein Plan planmäßig abgearbeitet wird, d. h. Ereignisse wie das Verlassen eines Tourstops oder die Ankunft bei einem Tourstop finden genau zu den für den Plan berechneten Zeitpunkten statt. Weiterhin wird angenommen, dass Vorgänge wie die Bearbeitung eines Tourstops oder die Fahrt von einem Tourstop zum nächsten Tourstop genau die dafür angegebene oder berechnete Zeitspanne in Anspruch nehmen. Für die Planungshorizonte wird eine zeitliche Dauer angesetzt, die in etwa einem Arbeitstag entspricht. Um die Dauer eines Tests innerhalb eines vernünftigen zeitlichen Rahmens zu halten, muss die simulierte Zeit schneller als die reale Zeit ablaufen. Der Standardwert für den Beschleunigungsfaktor beträgt 60, d. h. eine Sekunde Realzeit entspricht 60 Sekunden im simulierten Zeitablauf. Ein Test mit einem Planungshorizont von der Länge eines Arbeitstags mit acht Stunden dauert somit acht Minuten. Der Beschleunigungsfaktor kann während des Tests vom Benutzer verringert bzw. erhöht werden.

Die Transportfahrzeuge halten in Instanzvariablen je eine Tour Connection sowie einen Tourstop. Die aktuelle Position eines Transportfahrzeugs ist entweder der Mittelpunkt der Tour Connection oder die Position des Tourstops. Eine systeminterne Uhr, realisiert durch einen

Thread, simuliert den Zeitablauf des Planungshorizonts, indem periodisch der simulierte Zeitpunkt innerhalb des Planungshorizonts um den Beschleunigungsfaktor inkrementiert wird. Nach der Inkrementierung wird der neue Zeitpunkt innerhalb des Planungshorizonts an das Objekt übermittelt, das den Plan repräsentiert. Das Plan–Objekt setzt bei allen Fahrzeugen den neuen Zeitpunkt innerhalb des Planungshorizonts. Diese entscheiden daraufhin, ob sie ihre aktuelle Postion wechseln und setzten gegebenenfalls den neuen Zustand des Objekts, an dem sie sich gerade befinden (Tour Connection oder Tourstop) sowie den neuen Zustand des Objekts, das ihre neue Position repräsentiert.

5.2 Das MRVRP–Format

5.2.1 MRVRP–Plaintext–Format

Das MRVRP–Plaintext Format ermöglicht die einfache Erstellung kleinerer Instanzen mit einem Texteditor. Das Format wird anhand der folgenden Instanz mit zehn Kunden beschrieben:

```
'MRVRP 11'
time-type       HH-MM
customer-amount  10
request-amount   09

vehicle
capacity amount distance-cost time-cost
      50     3           1     1000

depot-model      SafetyLockDepotModel
lock-capacity    3
dep-service-time 00:20
arr-service-time 00:40

id  name             x     y    tw-start  tw-end

00  'depot'         25.0  25.0  08:00  22:30

01  'Haspa 01'      24.2  42.3
02  'Haspa 02'      10.0  34.2
03  'Penny 03'      13.5  23.7
```

```
04  'Penny 04'      31.5  17.5
05  'Penny 05'      13.0  14.8
06  'Penny 06'      35.0  37.3
07  'Penny 07'      18.0  27.5
08  'Post 08'       23.4  29.5
09  'Post 09'        1.4  35.4
10  'Post 10'       21.3  12.8

request constr.  location
num type    cap. id  tw-start tw-end serv-time

01  P   C2   20   3  08:00  18:00  00:10
02  P   C2   20   4  08:00  18:00  00:10
03  D   C2   20   5  08:00  18:00  00:10
04  D   C2   20   6  08:00  18:00  00:10
05  D   C2   20   7  08:00  18:00  00:10
07  P   C2   12   2  08:00  18:00  00:10
08  D   C2    9   9  08:30  17:00  00:05
09  PD  C2   15  10  08:00  19:00  00:05
                   1  09:00  18:00  00:10
```

Die erste Zeile enthält den Namen der Instanzdatei, in einfachen Hochkommata eingeschlossen. In der zweiten Zeile ist der Zeittyp definiert. Mögliche Werte sind zur Definition sind *HH-MM* für die Zeitabgabe in Stunden und Minuten sowie *int_30* für die Zeitangabe in Integer–Werten. Die Integer–Werte für Zeitangaben werden bei Definition von *int_30* in Zeitangaben in Stunden und Minuten umgewandelt. Dadurch wird die Übernahme von Daten aus Instanzen im Solomon bzw. Ropke und Pisinger Format vereinfacht. In Zeile drei und vier sind die Anzahl der Kunden und der Requests angegeben. Nach einer Leerzeile erfolgt die Definition der Fahrzeugkapazität, Anzahl der verfügbaren Fahrzeuge sowie die Werte der Adaptionsfaktoren der Testtreiber zur Ermittlung der Distanzen und Fahrtzeiten. Wiederum nach einer Leerzeile erfolgt die Definition der Schleuse. Als Depotmodell können das einfache Depotmodell ohne Schleuse durch Angabe des Schlüsselworts *SimpleDepotModel* oder das Depotmodell mit Schleuse durch Angabe des Schlüsselworts *SafetyLockDepotModel* gewählt werden. Wird als Depotmodell das Modell *SimpleDepotModel* gewählt, sind für die Schleuse keine weiteren Angaben zu machen. Im anderen Fall erolgt die Definition der Schleusenkapazität sowie der Bearbeitungszeiten für die Fahrzeuge beim Verlassen des Depots bzw. der Anfahrt zum Depot. Die Werte sind in demjenigen Zeitformat anzugeben, das bei “time-type” als Zeittyp angegeben ist. Danach erfolgt die Angabe der Tourstops. Dabei ist für jeden Tourstop die ID (als Integer–Wert anzugeben), der in Hochkommata eingeschlossene Identifier bzw. Name, sowie die Koordinaten im zweidimensionalen euklidischen Raum anzugeben.

Das Depot wird grundsätzlich als erster Tourstop mit Angabe des Zeitfensters des Depots (üblicherweise das Zeitintervall des Planungshorizonts) angegeben. Zuletzt erfolgt die Angabe der Requests. Dabei sind für jeden Request folgende Angaben einzutragen: Nummer des Requests und der Requesttyp. Für den Requesttyp stehen die Werte *P* für Pickup–Request, *D* für Delivery–Request und *PD* für Pickup–und–Delivery–Request zur Auswahl. Danach erfolgt die Angabe des Constraints für den Request. Mögliche Werte sind hier *C0* für Tourstops, bei denen die Mindestbesatzungsstärke des Fahrzeugs nicht relevant ist, *C2* für Tourstops, bei denen die Mindestbesatzungsstärke zwei Mann beträgt und *C3* für Tourstops, bei denen die Mindestbesatzungsstärke drei Mann beträgt. Nach der Angabe der Constraints erfolgt die Angabe der Menge der abzuholenden oder auszuliefernden Werte, die ID des Tourstops, das Zeitfenster, innerhalb dessen die Bearbeitung des Requests beim Tourstop begonnen werden soll, sowie die Servicezeit, d. h. die für die Bearbeitung des Requests beim Tourstop benötigte Zeit. Bei Pickup–und–Delivery–Requests sind anschließend noch die ID des Delivery–Tourstops (der erste angebene Tourstop ist hier immer der Pickup–Tourstop), sowie das Zeitfenster zur Bearbeitung und die Servicezeit des Delivery–Tourstops anzugeben.

5.2.2 MRVRP–XML–Format

Das MRVRP–XML–Format ist von der Abfolge her, in der die Daten zur Definition der jeweiligen Instanz angebenen werden, ähnlich aufgebaut wie das MRVRP–Plaintext–Format. Der einzige Unterschied besteht dabei darin, dass das Depotmodell nicht am Anfang der Datei, sondern zusammen mit den übrigen Daten des Depots angegeben wird. Bei der Eingabe von Instanzdateien im MRVRP–XML–Format werden die Dateien und die in ihnen enthaltenen Werte vom Parser anhand eines XML–Schema auf ihre Gültigkeit hin überprüft. Dabei finden folgende Prüfungen statt:

1. Überprüfung auf Einhaltung des am Anfang der XML–Datei festgelegten Zeitformats. Im gegenwärtigen Entwicklungsstand des Prototypen wird nur das Zeitformat *time* verwendet, d. h. der Planungshorizont muss innerhalb eines Tages liegen.
2. Überprüfung auf Einhaltung des am Anfang der XML–Datei festgelegten Formats für die Koordinaten der Stops. Im gegenwärtigen Entwicklungsstand des Prototypen wird nur das Koordinatenformat *euc2D* (Koordinaten im euklidischen zweidimensionalen Raum) unterstützt.
3. Überprüfung der Werte auf Verwendung des richtigen Datentyps bei der Angabe von Stop–ID, Koordinaten des Stops, Zeitfenster, Servicezeit und Kapazität.
4. Überprüfung der Stop–IDs bei der Angabe von Requests. Es dürfen bei der Angabe von Requests nur IDs von Stops auftreten, die zuvor mit ihren Koordinaten angegeben wurden.

5.3 Generator und Parser für Instanzdateien und Requests

Um den Prototypen mit Instanzen von verschiedenen Benchmarks zu testen, stehen Parser zur Verfügung, die die Instanzen des jeweiligen Benchmarks in Instanzen im MRVRP–XML–Format umwandeln. Abbildung 5.2 zeigt das Klassenmodell der Instanz–Parser und des MRVRP–Instanz–Generators.

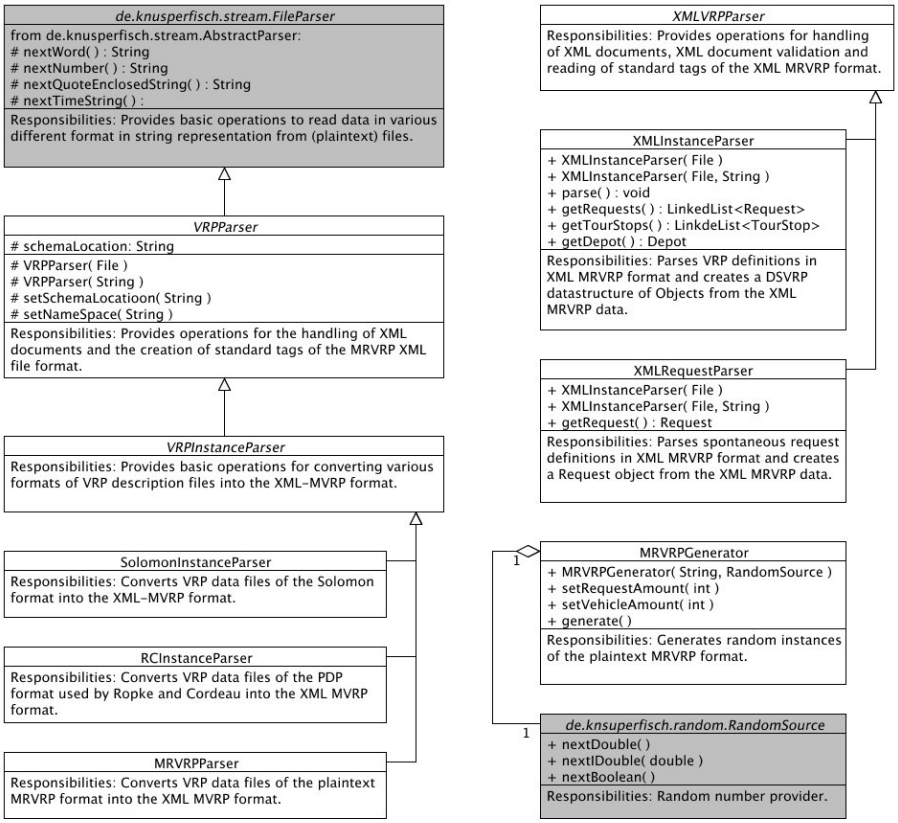

Abbildung 5.2: Klassen der Instanz–Generatoren/Parser

Von der Klasse *VRPInstanceParser* sind Parser abgeleitet, die Instanzen des Solomon Benchmarks (Klasse *SolomonInstanceParser*), des Benchmarks von Ropke und Cordeau (Klasse *RCInstanceParser*) sowie von Instanzen im MRVRP–Plaintext–Format in das MRVRP–XML–Format umwandeln. Von der Klasse *XMLVRPParser* ist der Parser *XMLInstanceParser* abgeleitet, der Instanzen im MRVRP–XML–Format parst und in Java–Objekte umwandelt. Dieser Parser wird bei der Eingabe von Instanzen in den Prototypen benutzt. Der ebenfalls von der Klasse *XMLVRPParser* abgeleitete Parser *XMLRequestParser* dient zum Parsen von spontanen Requests. Zur Generierung von MRVRP–Instanzen im MRVRP–Plaintext–Format dient der Generator *MRVRPGenerator*. Klassen, die Bestandteil des SAF sind, sind grau hinterlegt.

Das Zusammenspiel der Parser ist in Abbildung 5.3 veranschaulicht. Beim Einlesen einer Konfiguration für einen Testlauf ruft das Objekt der Klasse *DSVRP*, das das Optimierungsproblem darstellt, den *VRPXMLParser* auf, um die entsprechende Instanzdatei einzulesen. Spontane Requests, d. h. Request, die während des Planungshorizonts auftreten, werden durch den *XMLRequestParser* an das DSVRP übergeben.

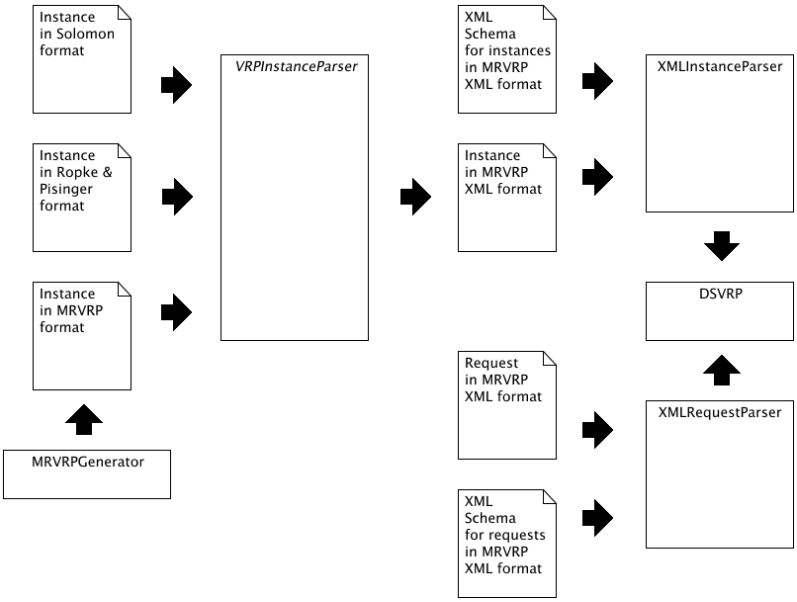

Abbildung 5.3: Datenfluss beim Generieren und Parsen von Instanz–Dateien

5.4 Algorithmus zur Optimierung

5.4.1 Allgemeiner Aufbau

Der allgemeine Aufbau des Algorithmus folgt dem von PISINGER UND ROPKE beschriebenen Aufbau des Frameworks ALNS [vgl. Pisinger und Ropke, 2007, S.7], wobei jedoch nicht alle Bestandteile des ALNS implementiert sind. ALNS basiert u. a. auf dem *Ruin and Recreate* Paradigma [vgl. Pisinger und Ropke, 2007, S.7], bei dem eine größere Anzahl an Tourstops entfernt und wieder eingefügt wird.

Als äußeres Framework kommt Simulated Annealing zum Einsatz. Der Algorithmus hält Referenzen auf drei Pläne: dem aktuellen Plan des Systems, der das bisher bekannte Minimum (*Overall Minimum Plan*) im Lösungsraum darstellt, der aktuellen Lösung des Algorithmus

(*Current Minimum Plan*), deren Nachbarschaft durch den Algorithmus durchsucht wird sowie der in der aktuellen Iteration erzeugten Lösung (*Current Plan*), die eine Lösung aus der Nachbarschaft des Current Minimum darstellt.

Der Ablauf einer Iteration ist in Abbildung 5.4 dargestellt: zunächst wird ein Entfernungsoperator und ein Einfügeoperator ausgewählt. Im gegenwärtigen Stand des Prototypen erfolgt die Auswahl nicht wie bei ALNS durch Gewichtung anhand der bisherigen Performanz der entsprechenden Operatoren, sondern nach in der Konfiguration des Algorithmus festgelegten Auswahl–Wahrscheinlichkeiten für die jeweiligen Operatoren.

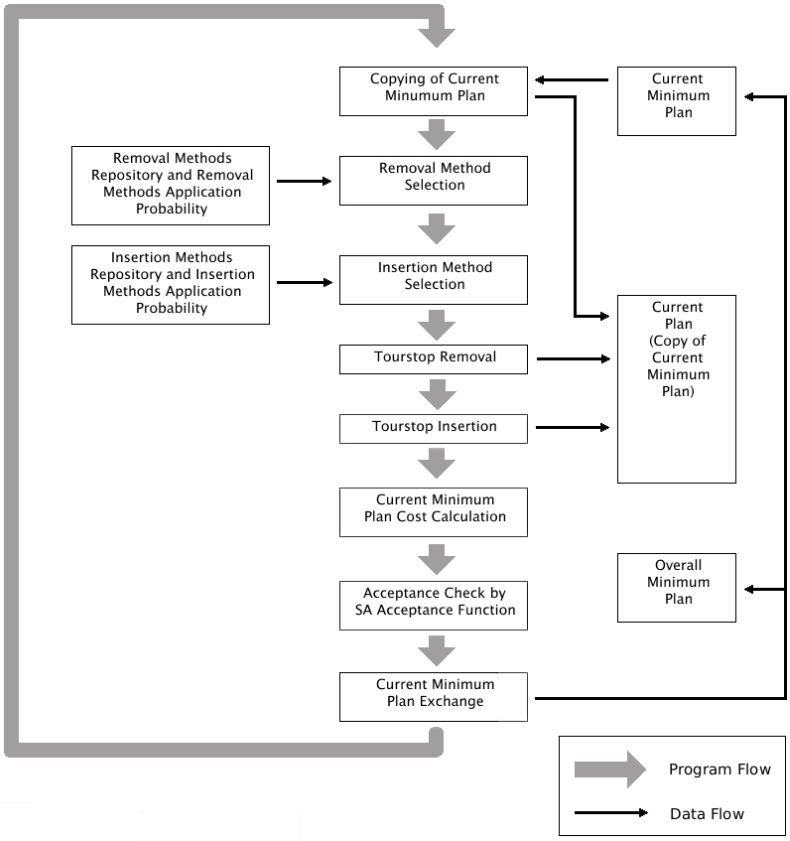

Abbildung 5.4: Ablauf einer Iteration des Optimierungsalgorithmus

Der Current Plan wird erzeugt, indem eine Kopie des Current Minimum Plan erstellt wird und aus dieser Kopie eine Anzahl an Tourstops entfernt und wieder eingefügt wird. Durch die Bewertungsfunktion werden die Kosten des neu erzeugten Plans errechnet. Anhand dieser Kosten und durch die Akzeptanzfunktion des Simulated Annealing wird entschieden, wie weiter zu verfahren ist. Ist der Current Plan schlechter als der Current Minimum Plan, wird durch die Akzeptanzfunktion entschieden, ob der Current Plan den Current Minimum Plan ersetzt. Ist der Current Plan besser als der Current Minimum Plan, wird der Current Minimum

Plan durch den Current Plan ersetzt. Ist der Current Plan besser als der Overall Minimum Plan, wird der Overall Minimum Plan durch den Current Plan ersetzt.

5.4.2 Konstruktionsheuristik

Die im Prototypen verwendete Konstruktionsheuristik ist vom Grundprinzip her einfach aufgebaut. Bei der Erzeugung einer Tour T_i wird von allen nicht gerouteten Tourstops V_r des Plans derjenige Tourstop v_m ausgewählt und aus V_r entfernt, der den größten Abstand zum Depot v_0 hat. Mit diesem wird die Anfangstour (v_0, v_m, v_0) erzeugt. Danach werden iterativ aus V_r derjenige Tourstop v_l sowie aus T_i diejenige Tour Connection c_{sd} ausgewählt, bei deren Einfügung von v_l in c_{sd} die Kosten der Tour T_i minimal erhöht werden. Dabei wird v_l aus V_r entfernt. Dies wird solange iterativ wiederholt, bis entweder V_r leer ist oder kein Tourstop aus V_r in T_i eingefügt werden kann. Auf diese Weise werden nacheinander die Touren des Plans erzeugt, bis V_r leer ist. Abbildung 5.5 veranschaulicht den Vorgang. Die Konstruktionsheuristik wird im Folgenden mit *Max–First–Then–Min Constructor* bezeichnet.

Die besondere Randbedingung der jeweiligen Mindestbesatzung zur Bearbeitung eines Requests einer Kundenart wird dadurch gehandhabt, dass zuerst alle Tourstops, die den Constraint C3 besitzen (Tourstops der Kundenart “Geldautomat”) geroutet werden. Danach werden alle Tourstops, die den Constraint C2 besitzen (Tourstops aller Kundenarten außer der Kundenart “Geldautomat”), geroutet. Dadurch soll vermieden werden, dass in Touren mit C3–Tourstops viele C2–Tourstops geroutet werden. Ein Routing von C2–Tourstops in C3–Touren kann allerdings in der Phase der Optimierung durch den Optimierungsalgorithmus stattfinden.

Es besteht grundsätzlich die Möglichkeit, dass bei der Erzeugung eines Plans durch eine Konstruktionsheuristik nicht alle Tourstops in eine Tour geroutet werden können. Um zu vermeiden, dass die Konstruktionsheuristik in eine Endlosschleife gerät, in der immer wieder erfolglos versucht wird, eine Menge an restlichen, nicht gerouteten Tourstops zu routen, wird die Liste der nicht gerouteten Tourstops V_r beobachtet. Verringert sich während einer Iteration des Konstruktionsalgorithmus die Anzahl der Tourstops in V_r nicht, ist dies ein Anzeichen für das Auftreten der eben beschriebenen Situation und die Konstruktionsheuristik sollte abgebrochen werden. Diese Prüfung ist jeodch im gegenwärtigen Stand der Implementation der Konstruktionsheuristik nicht enthalten.

Die Idee hinter der Entwicklung des Max–First–Then–Min Konstruktor ist, dass auf dem Weg zu einem weit entfernten Tourstop diejenigen Tourstops mit abgearbeitet werden sollten, die in der Nähe zum Weg des weit entfernten Tourstops liegen. Durch Zusammenfassen dieser Tourstops zu einer Tour kann dabei auf einfache Weise ein Clustering erreicht werden.

Abbildung 5.5: Konstruktion einer Tour durch den Max–First–Then–Min Konstruktor

5.4.3 Modifikationsoperatoren

Für die Modifikation eines bestehenden Plans wurden je zwei Entfernungs– und Einfüge–Operatoren implementiert, die jeweils eine in der Konfiguration des Algorithmus festgelegt Anzahl von Tourstops entfernen bzw. einfügen. Der *Most Expensive Removal Modifier* entfernt eine Anzahl der Tourstops, die die meisten Kosten in der Tour verursachen, d. h. es werden die Tourstops entfernt, bei deren Entfernung die Minimierung der Kosten der Tour maximal werden. Der *Random Removal Modifier* entfernt zufallsgesteuert eine Anzahl von Tourstops. Der *Min Cost Insertion Modifier* fügt eine Anzahl von Tourstops in einen Plan an die Stellen ein, bei der durch Einfügung des jeweiligen Tourstops die Kosten des Plans minimal erhöht werden. Der *Random Insertion Modifier* fügt eine Anzahl von Tourstops zufallsgesteuert in einen Plan ein. Alle Operatoren führen die jeweilige Modifikation unter Beachtung aller Randbedingungen wie Kapazitätsbeschränkungen oder Zeitfenster durch.

5.4.4 Berechnung des Time Slack

Zur Berechnung des Time Slack wird die in Kapitel 2.4.4.2 (Verwendung von Zeitpuffern) von CORDEAU U. A. angegebene Formel 2.22 verwendet.

5.4.5 Berechnung des Capacity Slack

Die Berechnung des Capacity Slacks, d. h. der restlichen Kapazität des Transportfahrzeugs, die für weitere Requests genutzt werden kann, muss abhängig von der Art des Requests erfolgen. Bei Pickup–Requests wird die für den jeweiligen Request benötigte Fahrzeugkapazität vom Anfang der Tour bis zum Tourstop, an dem die Werte abgeliefert werden, genutzt. Bei Delivery–Requests wird die für den jeweiligen Request benötigte Fahrzeugkapazität vom Tourstop, an dem die Werte abgeholt werden, bis zum Ende der Tour genutzt. Bei Pickup–and–Delivery–Requests wird die für den jeweiligen Request benötigte Fahrzeugkapazität vom Tourstop, an dem die Werte abgeholt werden, bis zum Tourstop, an dem die Werte abgeliefert werden, genutzt.

Gegeben sei eine Tour $T = v_0, v_1, \ldots, v_q$. Die Ladung l_{tv_i} eines Transportfahrzeugs t nach Bearbeitung des Tourstops v_i beträgt

$$l_{tv_i} = \sum_{0 \leq j \leq i} (c_j \cdot p_j - c_j \cdot d_j) \tag{5.1}$$

wobei c_j die zur Bearbeitung des Requests benötigte Fahrzeugkapazität darstellt. p_j und d_j sind binäre Entscheidungsvariablen. p_j nimmt den Wert 1 an, wenn v_j ein Pickup–Tourstop ist, ansonsten den Wert 0. d_j nimmt den Wert 1 an, wenn v_j ein Delivery–Tourstop ist, ansonsten den Wert 0.

Der Pickup–Slack $St_{p(s)}$ beschreibt die restliche Kapazität des Transportfahrzeugs t, die für einen neuen Pickup–Request zur Verfügung steht, bei dem die Werte beim Tourstop s abzuholen sind, und ist definiert durch:

$$St_{p(s)} = \min_{s \leq i \leq q} \{c_t - l_{tv_i}\} \tag{5.2}$$

wobei c_t die maximale Kapazität des Fahrzeugs darstellt. Der Delivery–Slack $St_{d(s)}$ beschreibt die Kapazität, die für einen neuen Delivery–Request zur Verfügung steht, bei dem die Werte beim Tourstop s abzuliefern sind und ist definiert durch:

$$St_{d(s)} = \min_{0 \leq i \leq s} \{c_t - l_{tv_i}\} \tag{5.3}$$

Der Pickup–and–Delivery–Slack $St_{pd(n,k)}$ beschreibt die Kapazität, die für einen neuen Pickup–and–Delivery–Request zur Verfügung steht, bei dem die Werte beim Tourstop n abzuholen und beim Tourstop k abzuliefern sind und ist definiert durch:

$$St_{pd(n,k)} = \min_{n \leq i \leq k} \{c_t - l_{v_i}\} \tag{5.4}$$

5.4.6 Bewertungsfunktion

In der Bewertungsfunktion (Objective Function) werden die Kosten der einzelnen Touren eines Plan berechnet, die aufsummierten Kosten der einzelnen Touren ergeben die Gesamtkosten eines Plans. Die Berechnung der Kosten einer Tour (Wert der Objective Function für eine Tour) ist abhänging vom Benchmark bzw. dem Format der Instanz, für die die Kosten berechnet werden. Als Kosten einer Tour werden die Kosten des Fahrzeugs, das die Tour abfährt, berechnet. Die Kosten setzen sich dabei aus den folgenden Teilkosten zusammen:

1. Kosten des Fahrzeugs für die Länge der Tour, d. h. die zu überwindende Distanz.
2. Kosten des Fahrzeugs für die Zeitspanne, die benötigt wird, um eine Tour abzufahren.
3. Kosten der Service Constraints des Fahrzeugs für die Länge der Tour.
4. Kosten der Service Constraints des Fahrzeugs für die Zeitspanne, die benötigt wird, um eine Tour abzufahren.

Durch die Definition von Kosten für Service Constraints und das Miteinbeziehen der Kosten der Service Constraints in die Berechnung der Kosten einer Tour kann auf einfache Weise die besondere Randbedingung der verschiedenen Kundenarten, für die jeweils verschiedene Mindesbesatzungsstärken der Transportfahrzeuge zur Bearbeitung eines Requests des jeweiligen Kunden nötig sind, abgebildet werden.

Als Beispiel soll hier der Service Constraint C3 dienen, der an Tourstops der Kundenart “Geldautomat” sowie an Fahrzeuge, deren Touren Geldautomaten enthalten können, vergeben wird. Zur Bearbeitung eines Requests eines Stops vom Typ “Geldautomat” muss das Fahrzeug eine Mindestbesatzungsstärke von drei Mann aufweisen. Der Stundenlohn eines Besatzungsmitglieds wird mit 10 Euro angenommen. Der Service Constraint C3 enthält als Kosten die Lohnkosten für drei Besatzungsmitglieder pro Zeiteinheit. Das System arbeitet intern mit der Zeiteinheit von einer Millisekunde. Die Kosten für eine Zeiteinheit des Systems betragen somit $3 \cdot 10/(60 \cdot 60 \cdot 1000)$. Für den Service Constraint C3 sind die Kosten einer Distanzeinheit mit 0 definiert.

5.5 Benutzerschnittstelle

5.5.1 Allgemeiner Aufbau

Abbildung 5.6 zeigt die Benutzerschnittstelle des Prototypen. Zur Optimierung einer VRP–Instanz muss zunächst eine Konfigurationsdatei erstellt werden, die die zu verwendende Zufallszahlenquelle, die zu optimierende Probleminstanz, die Parameter des Optimierungsalgorithmus und den Typ des Generators zur Erstellung von Startlösungen (gegebenfalls ebenfalls mit Angabe der Parameter) enthält. Konfigurationsdateien können über die Auswahlfolge der Menupunkte *Configuration*, *load from File*, *Testrun* ausgewählt und geladen werden.

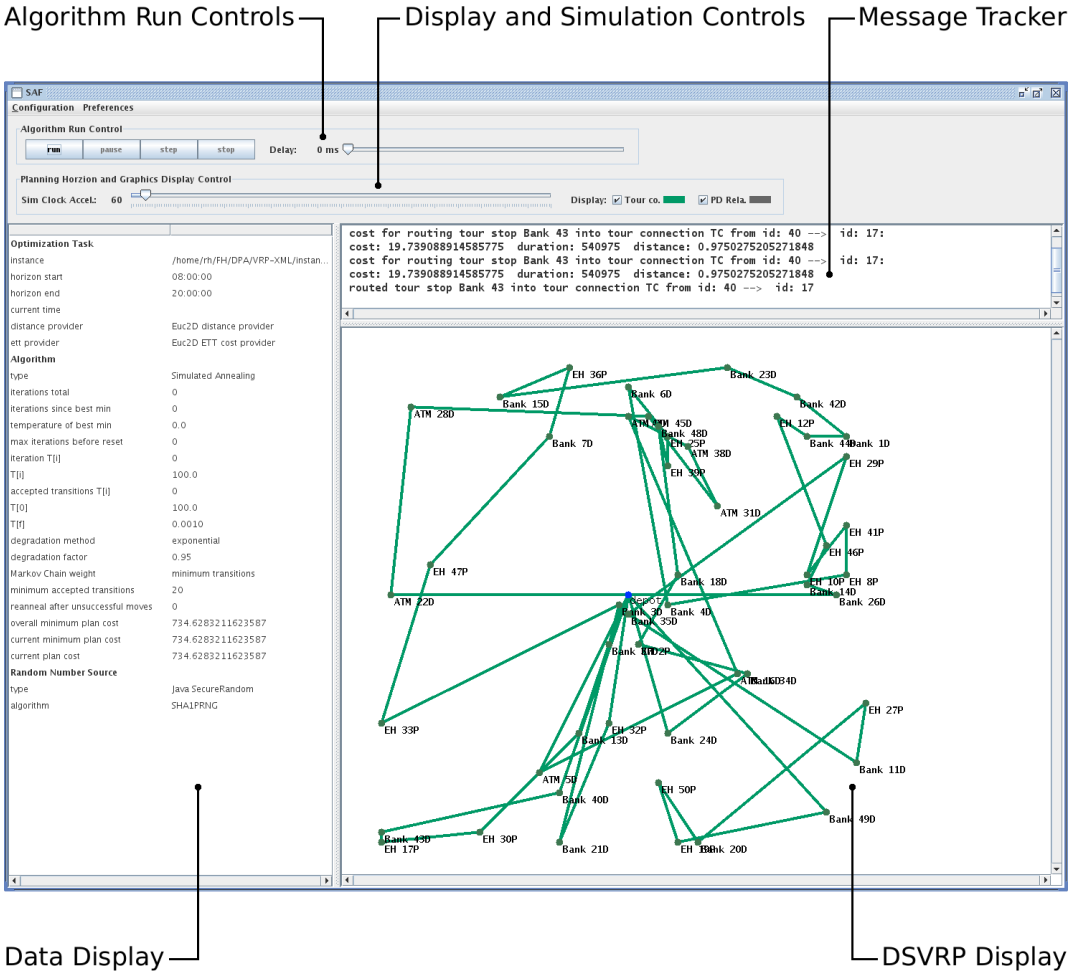

Abbildung 5.6: Benutzerschnittstelle des Prototypen

Die Benutzerschnittstelle enthält je ein Kontrollelement zur Steuerung des Ablaufs der Opti-

mierung sowie der Simulation des Planungshorizonts. Das Steuerungelement *Algorithm Run Control* dient zur Steuerung des Ablaufs der *a–priori* Optimierung. Das Element enthält vier Buttons zur Steuerung der automatischen Optimierung durch das System: der Button *Run* startet den Optimierungsalgorithmus. Mit dem Button *Pause* kann ein laufender Optimierungsalgorithmus unterbrochen werden, um z. B. die aktuelle Instanz interaktiv zu optimieren. Eine Unterbrechung des Optimierungsalgorithmus kann durch erneute Auswahl des Buttons *Run* aufgehoben werden. Mit Auswahl des Buttons *Step* können im unterbrochenen Zustand des Optimierungsalgorithmus einzelne Iterationen des Optimierungsalgorithmus ausgelöst werden. Durch Auswahl des Buttons *Stop* wird die *a–priori* Optimierungsphase beendet und die Simulation des Planungshorizonts gestartet. Das Algorithm Run Control enthält einen Slider, mit dem die Ablaufgeschwindigkeit des Optimierungsalgorithmus eingestellt werden kann. Dadurch ist es möglich, bei schnell ablaufenden Algorithmen den Verlauf der Optimierung besser beobachten zu können.

Die *Display and Simulation Controls* enthalten einen Slider, mit dem die Ablaufgeschwindigkeit der Simulation des Planungshorizonts eingestellt werden kann. Dabei ist es auch möglich, den Ablauf der Simulation anzuhalten und später wieder fortzusetzen. Mit dem Slider können für den Beschleunigungsfaktor des Ablaufs der simulierten Zeit Werte von 0 (Simulation wird angehalten) bis 1800 (1 Sekunde Realzeit entspricht einer halben Stunde des simulierten Planungshorizonts) eingestellt werden. Neben dem Slider befinden sich zwei Checkboxen, mit denen die Anzeige von graphischen Elementen des *DSVRP Displays* ein– und ausgeblendet werden können. Das Ausblenden der Tour Connections ermöglicht, insbesondere bei größeren Instanzen, eine bessere Übersicht über die geographische Verteilung der Tourstops. Zur Anzeige von PD–Requests dienen graphische Elemente des Typs *PD–Relation*, die eine gestrichelte Linie vom Pickup–Tourstop eines PD–Requests zum Delivery–Tourstop darstellen. Diese graphischen Elemente können ebenfalls ein– und ausgeblendet werden.

Das *DSVRP Display* stellt die zu optimierende Instanz als Graph dar und dient zugleich als Schnittstelle zur interaktiven Optimierung durch den Dispatcher. Der *Message Tracker* dient zur Ausgabe von Textmeldungen an den Dispatcher. Die Klasse des Message Trackers ist als das von GAMMA U. A. beschriebene Singleton [vgl. Gamma u. a., 1994, S.127] implementiert. Dadurch kann dieser von jeder Stelle des Systems ohne Referenz benutzt werden, allerdings wird der Message Tracker hauptsächlich vom User Action Router dazu verwendet, Informationen über Elemente der Instanz wie z. B. Tourstops und Statusinformationen während der interaktiven Veränderung an den Dispatcher auszugeben. Das *Data Display* dient zur Anzeige von numerischen Werten. Dies sind u. a. die Parameter des Optimierungsalgorithmus sowie die Werte der Kosten von Startlösung und der bisher besten gefundenen Lösung.

5.5.2 Schnittstelle für die Interaktion des Dispatchers

In Abbildung 5.7 ist das Klassenmodell des Displays und der Interaktionsschnittstelle dargestellt. Das *DSVRP Display* enthält zwei Arten von Anzeige–Elementen: aktive Elemente dienen zur Anzeige von Elementen der Instanz (z. B. Tourstops) und als Schnittstelle für die Interaktion des Dispatchers, nichtaktive Elemente dienen nur zur Anzeige.

Die aktiven Elemente werden im Klassenmodel durch die Klasse *Interactive VRP Element Shape* repräsentiert. Davon abgeleitet sind die graphischen Elemente zur Anzeige von Tourstops (Klasse *Tour Stop Shape*), Tour Connections (Klasse *Tour Connection Shape*) und Transportfahrzeugen (Klasse *Vehicle Shape*). Jedes Objekt dieser Klassen hält jeweils eine Referenz auf das Objekt, zu dessen graphischer Darstellung es im Display dient und ist zugleich Beobachter dieses Objekts. So hält z. B. ein Tour Stop Shape immer eine Referenz auf den Tour Stop, den es repräsentiert. Ändert ein Element der Instanz seinen Zustand, wird das entsprechende graphische Element zur Repräsentation davon benachrichtigt und kann daraufhin die Anzeige des zu repräsentierenden Elements entsprechend anpassen. Ein Beispiel dafür ist der Übergang des Zustands eines Tourstops vom Zustand *ROUTED* (Tourstop ist in einer Tour enthalten und seine Abarbeitung ist geplant) in den Zustand *APPROACHED* (das Transportfahrzeug zur Bearbeitung des Tourstops befindet sich in der Anfahrt zum Tourstop). Aktive Elemente besitzen ein durch Anklicken des Elements mit der rechten Maustaste abrufbares Popup–Menu, das wiederum eine Referenz auf den User Action Router hält. Dieser prüft, ob die vom Dispatcher aus dem Popup–Menü ausgewählten Aktionen zur Modifkation der gegenwärtigen Lösung eine gültige neue Lösung erzeugen. Fällt diese Prüfung positiv aus, wird die Aktion des Dispatchers durchgeführt und der gegenwärtige Plan entsprechend verändert. Über den *Message Tracker* erhält der Dispatcher Informationen über die Ausführbarkeit seiner beabsichtigten Interaktionen sowie Meldungen, die eine Modifikation des gegenwärtigen Plans bestätigen. Wird ein aktives Element mit der linken Maustaste angeklickt, werden im Message Tracker Informationen über das entsprechende Objekt angezeigt. Bei einem Tourstop sind dies u. a. das Zeitfenster, die Servicezeit und der geplante Zeitpunkt, zu dem das entsprechende Transportfahrzeug bei diesem Tourstop eintrifft.

Nichtaktive Elemente dienen zur Anzeige von zusätzlichen Informationen, die in bestimmten Fällen für den Dispatcher hilfreich sind. Dazu gehören die im vorhergehenden Abschnitt (Allgemeiner Aufbau) angeführten PD–Relations, (Klasse *PD Relation Shape* Im Klassenmodell) sowie Hilfslinien (Klasse *PD Move Hint Shape*), die beim Routen von PD–Requests dem Dispatcher Hilfestellung bieten. In Abbildung 5.7 grau hinterlegte Klassen mit Package–Namen stammen aus dem Framework SAF, grau hinterlegte Klassen ohne Package–Namen sind standardmässig in der Programmiersparche Java enthalten.

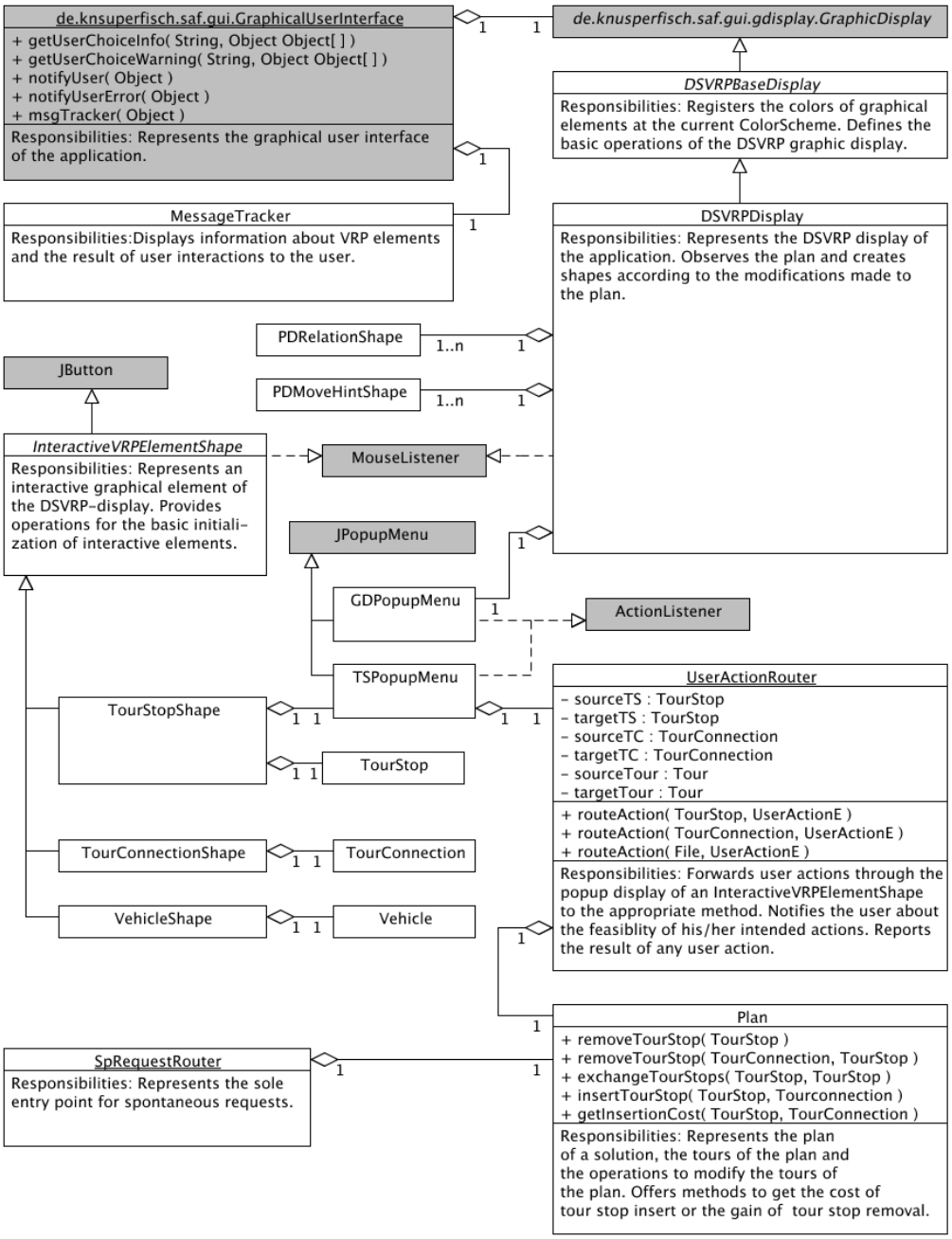

Abbildung 5.7: Klassenmodell des Displays und der Interaktionsschnittstelle

5.5.3 Anzeige von Instanzen

Die Zustände von Tourstops und Tour Connections werden durch verschiedene Farben angezeigt. Während der automatischen Optimierung durch den Algorithmus wird das DSVRP

Display automatisch aktualisiert, sobald der Algorithmus eine bessere Lösung (d. h. einen besseren Plan) als die gegenwärtige Lösung erreicht, d. h. im Display wird die gegenwärtige Lösung gegen die neue, bessere Lösung ausgetauscht.

5.5.4 Anzeige während der Simulation des Planungshorizonts

Während des simulierten Ablaufs des Planungshorizonts werden zusätzlich zur Anzeige der Instanz als Graph auch die Transportfahrzeuge, die eine Tour bearbeiten, angezeigt. Die Zustände eines Transportfahrzeugs (Befahren einer Tour Connection, Abwarten des Anfangszeitpunktes eines Tourstops, Be– und Entladen von Werten) werden ebenfalls durch verschiedene Farben dargestellt. Die Farben der einzelnen Zustände von Tourstops, Tour Connections und Transportfahrzeugen können im Menü *Preferences* den Bedürfnissen bzw. Vorlieben des Dispatchers angepasst werden.

Abbildung 5.8 zeigt den Zustand einer Instanz zu verschiedenen Zeitpunkten während des Ablaufs der Simulation des Planungshorizonts. Bild 1 zeigt die Instanz nach der *a–priori* Optimierung und vor Start des Planungshorizonts. Bild 2 zeigt das erste Transportfahrzeug (mit der ID 3) bei der Abarbeitung seiner Tour. Bereits abgefahrene Tour Connections und Tourstops sind grau dargestellt, die gegenwärtig vom Fahrzeug befahrene Tour Connection ist hellgrün dargestellt und der Tourstop, zu dem sich das Fahrzeug in der Anfahrt befindet, ist orange dargestellt. Zum Zeitpunkt, der in Bild 3 dargestellt ist, sind zwei weitere Fahrzeuge (mit den IDs 1 und 2) dabei, die ihnen zugeordneten Touren abzuarbeiten, das Fahrzeug mit der ID 3 hat seine Tour beendet und wartet vor der Depotschleuse auf Einlass. In Bild 4 haben die Fahrzeuge mit den IDs 1 und 2 ihre Touren fast abgearbeitet. In Bild 5 ist das letzte Fahrzeug (mit der ID 0) bei der Anfahrt zum ersten Tourstop seiner Tour zu sehen. Bei dieser Tour handelt es sich um eine Tour, die im Gegensatz zu den anderen Touren Geldautomaten als Tourstops enthält. Daher liegt diese Tour auch im Bereich einer anderen Tour, der des Fahrzeugs mit der ID 1. Bild 6 zeigt die Instanz gegen Ende des Planungshorizonts, nachdem alle Touren abgearbeitet sind.

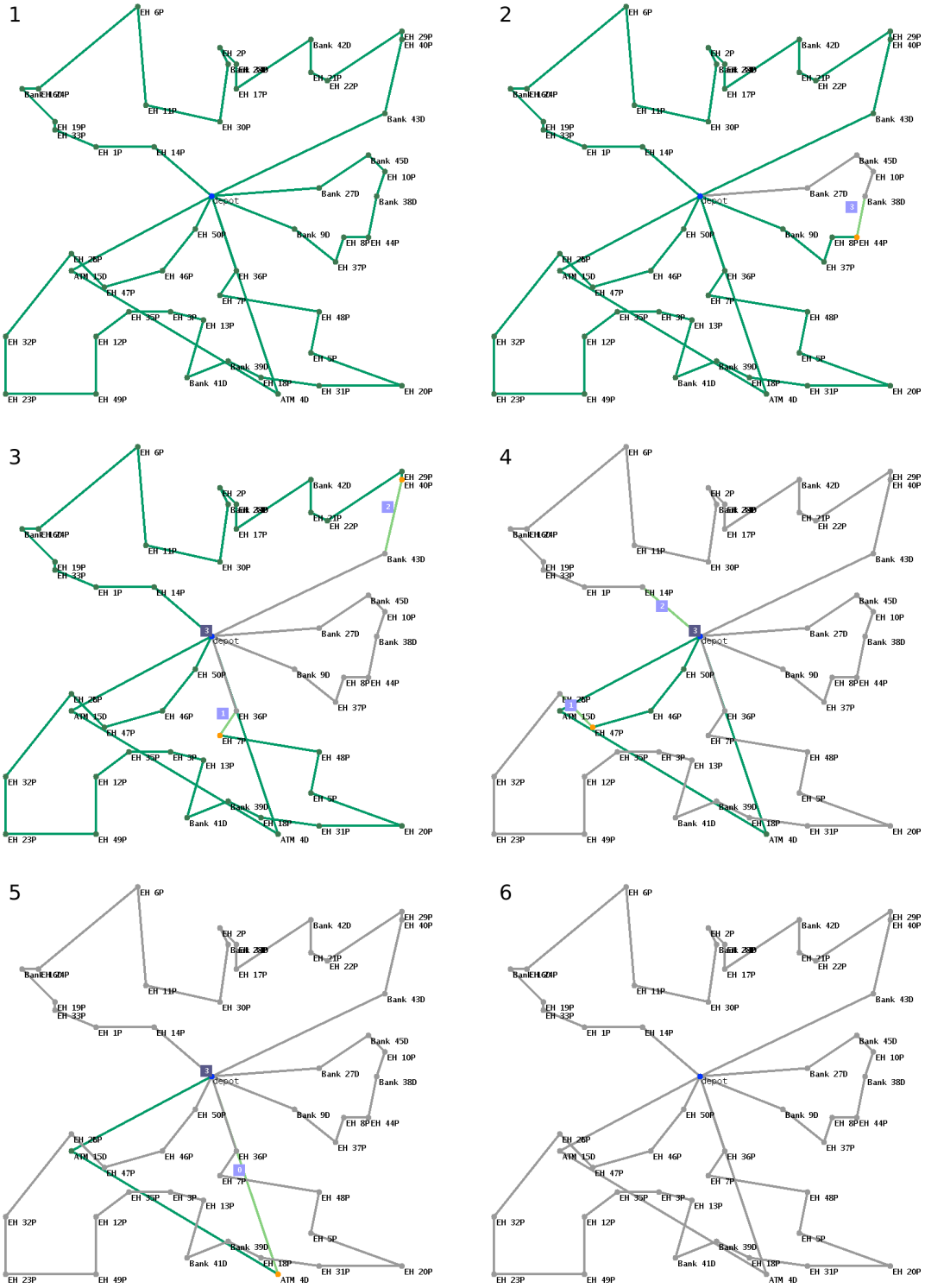

Abbildung 5.8: Anzeige des DSVRP während Ablauf des Planungshorizonts

5.6 Test

Bei den folgenden Tests wurde die Leistungsfähigkeit des Optimierungsalgorithmus bei der Optimierung von VRP ohne Interaktion eines Dispatchers getestet. Da beim gegenwärtigen Stand der Implementierung der Modifikationsoperatoren die Behandlung von PD–Requests nicht enthalten ist, wurden nur Tests mit Instanzen durchgeführt, die Pickup– und Delivery–Requests enthalten.

5.6.1 MRVRP Instanzen

Für die Tests wurden zwei Instanzen des MRVRP Formats generiert, die Instanz mrvrp 2.05 mit 50 Requests und die Instanz mrvrp 2.07 mit 100 Requests. Die Kennzahlen der Instanzen sind in der folgenden Tabelle 5.6.1 aufgeführt:

Anzahl der verfügbaren Fahrzeuge	10
Kapazität der Fahrzeuge	100
Prozentualer Anteil der Kundenarten	
Bank	0.15
ATM	0.20
Großhändler	0.25
Einzelhändler	0.40

Tabelle 5.1: Konfiguration zur Optimierung der Instanzen mrvrp 2.05 und mrvrp 2.07

Die Requests wurden jeweils mit den Standard–Zeitfenstern der jeweiligen Kundenart generiert. Für die Kundenart *Service* wurden keine Requests generiert, um die Kapazität der Fahrzeuge stärker zu belasten. Im Folgenden werden die Parameter der Konfiguration für die Testläufe mit den Instanzen mrvrp 2.05 und mrvrp 2.07 beschrieben: als Zufallszahlenquelle wird die Klasse Java Secure Random verwendet. Als Konstruktionsheuristik kommt der Max–First–Then–Min Constructor zum Einsatz. Die Starttemperatur des Simulated Annealing beträgt 100, die Endtemperatur 0,0001, bei exponentieller Verringerung der Temperatur mit dem Faktor 0,95. Die Länge der Markov–Ketten beträgt 20 akzeptierte Veränderungen des Plans. Bei jeder Iteration werden 10 Tourstops entfernt und wieder eingefügt. Die Auswahlwahrscheinlichkeiten für die Modifikationsoperatoren betragen 0,4 für den Random Removal Modifier, 0,6 für den Most Expensive Removal Modifier, 0,15 für den Random Insertion Modifier und 0,85 für den Min Cost Insertion Modifier.

Abbildung 5.9 zeigt die Instanz mrvrp 2.05 nach der Konstruktion der Startlösung (Bild 1) und nach 500 Iterationen des Optimierungsalgorithmus (Bild 2). Der Verlauf der Kosten der während der Optimierung generierten Lösungen für die Instanz mrvrp 2.05 ist in Abbildung 5.10 dargestellt.

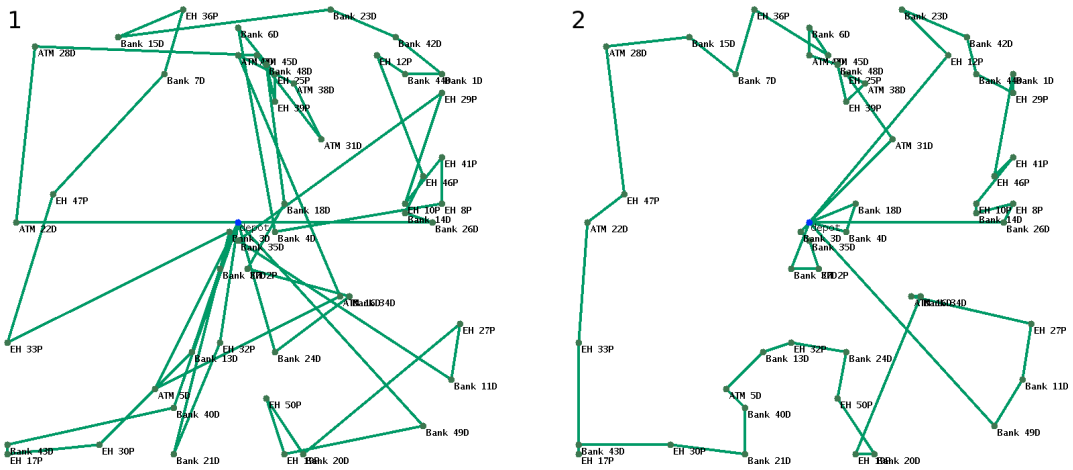

Abbildung 5.9: Instanz mrvrp 2.05 vor und nach der automatischen Optimierung

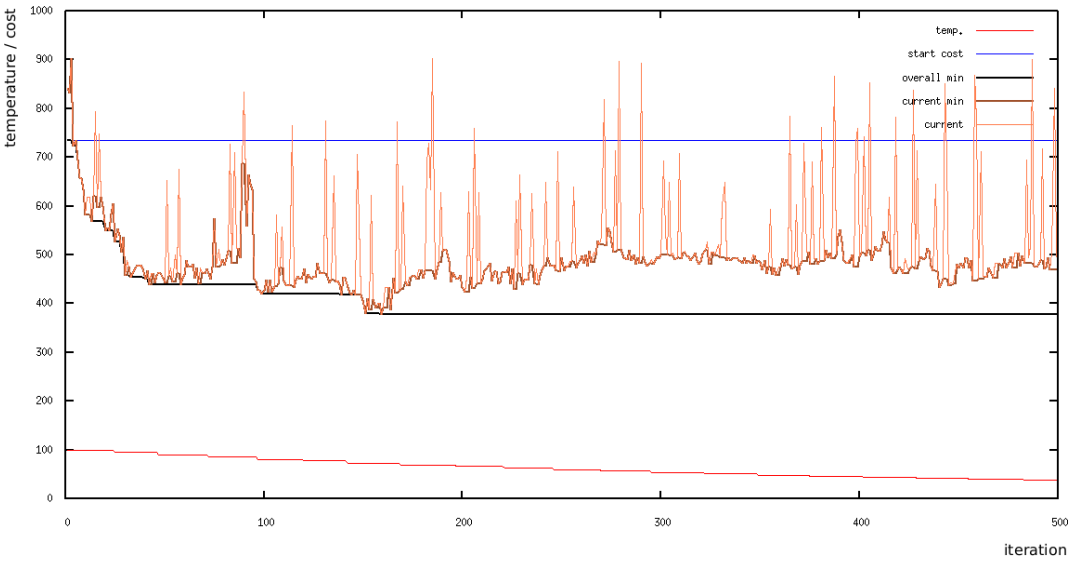

Abbildung 5.10: Verlauf der Kostenoptimierung, Instanz mrvrp 2.05

Abbildung 5.11 zeigt die Instanz mrvrp 2.07 nach der Konstruktion der Startlösung (Bild 1) und nach 500 Iterationen des Optimierungsalgorithmus (Bild 2). Der Verlauf der Kosten der während der Optimierung generierten Lösungen für die Instanz mrvrp 2.05 ist in Abbildung 5.12 dargestellt.

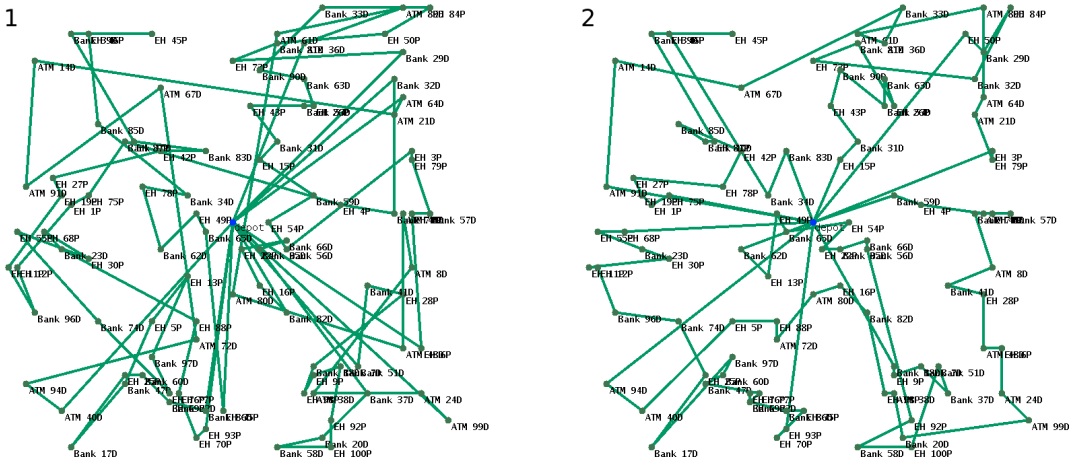

Abbildung 5.11: Instanz mrvrp 2.07 vor und nach der automatischen Optimierung

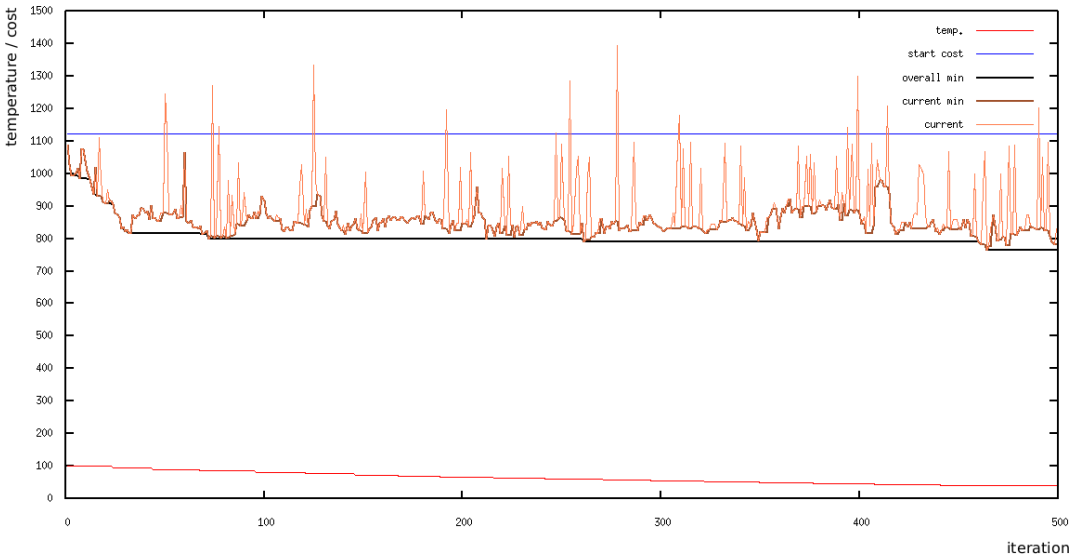

Abbildung 5.12: Verlauf der Kostenoptimierung, Instanz mrvrp 2.07

5.6.2 Bewertung der Testergebnisse

Das Annealing Schedule mit einem Temperatur–Verringerungsfaktor von 0,95 bei exponentieller Senkung der Temperatur und einer Länge der Markov–Ketten von 20 akzeptierten Iterationen führt sehr schnell zum Erreichen eines lokalen Minimums. Dies stellt in einem interaktiven System jedoch keinen Nachteil dar, da durch ein Eingreifen des Dispatchers der Algorithmus aus einem lokalen Minimum befreit werden kann.

Die beiden Bilder der Instanz in Abbildung 5.9 vor Start des Optimierungsalgorithmus sowie nach 500 Iterationen zeigen, dass der Algorithmus die Kosten der Lösung gut senkt. Die

Darstellung des Verlaufs der Kostenoptimierung (Abbildung 5.10) zeigt, dass nach ca. 150 Iterationen das Potential des Annealing Schedules weitgehend ausgeschöpft ist, eine weitere Verbesserung der Lösung wird innerhalb der 500 Iterationen nicht mehr erreicht. Bild 2 zeigt, dass durch eine Interaktion eines Dispatcher noch weitere Verbesserungen erreicht werden können.

Bei der Optimierung der Instanz mrvrp 2.07 fängt sich der Algorithmus schon nach ca. 30 Iterationen in einem lokalen Optimum, dass keine gute Lösung darstellt. Danach werden innerhalb der 500 Iterationen keine nennenswerten Verbesserungen erzielt. Der Verlauf der Kostenoptimierung zeigt, dass sich der Algorithmus nach Erreichen des lokalen Optimums nach ca. 30 Iteration nur wenig von der bisher erreichten besten Lösung entfernt. Bild 2 in Abbildung 5.11 lässt erahnen, dass der Algorithmus nur durch eine umreiche Modifikationen des Plans durch den Dispatcher aus dem lokalen Optimum zu befreien ist.

Da die Implementation des Prototypen sich als um einiges aufwendiger herausgestellt hat, als zu Beginn der Arbeit angenommen wurde, musste auf umfangreichere Tests des Systems verzichtet werden. Aus diesem Grund wurde das System lediglich mit zwei instanzen im MR-VRP Format getestet. Die Ergebnisse der Tests zeigen, dass das System grundsätzlich in der Lage ist, Probleminstanzen aus dem Anwendugsbereich zu optimieren. Zur Abschätzung der Leistungsgrenzen des Optimierungsalgorithmus im gegenwärtigen Stand der Implementation wären jedoch weitere Tests nötig. Dabei sollten insbesondere bei größeren Instanzen langsamere Annealing Schedules verwendet werden.

6 Zusammenfassung

Am Anfang dieser Arbeit stand die Idee, ein System zur interaktiven und dynamischen Planung und Optimierung von Touren zu konzipieren, das die besonderen Anforderungen an Planungssysteme im Werttransport–Bereich berücksichtigt. Dazu gehören die Planung von Schleusenbelegungszeiten sowie die Problematik der verschiedenen Kundenarten, die sich in der zur Bearbeitung eines Requests benötigten Mindestbesatzungsstärke eines Transportfahrzeugs unterscheiden. Ein menschlicher Dispatcher sollte durch das System bei der Planung und Optimierung von Touren unterstützt werden, durch die Interaktion des Dispatchers sollten die bisher dem Menschen vorbehaltenen Problemlösungkompetenzen im Bereich von Tourenplanung mit den Vorteilen maschinenbasierter Optimierung vereint werden. Durch die Fähigkeit des Systems, erstellte Pläne während der Ausführung an veränderte Umweltbedingungen, wie z. B. die Änderungen von Fahrtdauern aufgrund veränderter Verkehrsverhältnisse und an spontane Aufträge anzupassen, sollte eine Möglichkeit zu einer operativen Planung geschaffen werden, deren Ergebnisse insgesamt gegen Umwelteinflüsse robuster sind.

Zur Lösung der Problemstellung wurden zunächst die aus dem VRP–Bereich bekannten Verfahren zur Optimierung von VRP untersucht. Dabei wurde auch auf Lösungsansätze zur Behandlung dynamischer VRP eingegangen. Die Verfahren wurden hinsichtlich ihrer Eignung zur Anwendung auf die Problemstellung sowie ihrer Eignung zur Integration in ein interaktives System analysiert.

Weitere Punkte der Analyse waren die Untersuchung der bisher im VRP–Bereich bekannten Modelle zur Modellierung von Problemstellungen aus dem Logistik–Bereich und die Untersuchung der Anforderungen an ein System zur interaktiven Optimierung von Vehicle Routing Problemen.

Die Anwendbarkeit des erstellten Konzepts sollte anhand eines Prototypen verifiziert werden. Dazu sollten ausgewählte Teilfunktionen des Systems im Prototypen implementiert werden. Als Grundlage zur Implementation sollte dabei das in der Studienarbeit des Autors entwickelte Framework für stochatische Algorithmen SAF dienen.

6.1 Fazit

Durch die Wahl des ALNS als Framework für die Realisierung des Verfahrens zur automatischen Optimierung besteht die Möglichkeit der stufenweisen Implementation. Im gegenwärtigen Stand des Prototypen kann das Verfahren als “the Poor Man’s Large Neighbourhood Search” bezeichnet werden. Die Auswahl der Modifikationsoperatoren erfolgt festgelegt nach in der Konfiguration definierten Auswahlwahrscheinlichkeiten für die jeweiligen Operatoren. Es werden relativ einfache Operatoren verwendet, eine Rauschfunktion wie bei ALNS fehlt. Die durchgeführten Tests zeigen jedoch, dass selbst mit einfachen Operatoren bei guter Wahl der Parameter des Algorithmus gute Ergebnisse bei der Optimierung zu erzielen sind. Voraussetzung dafür sind jedoch etwas Erfahrung beim Einsatz der Metaheuristik Simulated Annealing und eine gewisses Einschätzungsvermögen, was die Leistungsfähigkeit der Modifikationsoperatoren anbelangt. Durch die Einfachheit der verwendeten Modifikationsoperatoren läuft eine Iteration sehr schnell ab. Insbesondere bei interaktiven Systemen ist dies von großem Vorteil.

Zur Modellierung der Randbedinungen der Problemstellung war die Entwicklung eines neuen Formats zur Beschreibung von VRP nötig. Mit der Spezifikation des MRVRP (Mixed Request VRP) steht nun ein Format zur Verfügung, das alle Arten von Requests (Pickup–Request, Delivery–Reuests, PD–Requests) berücksichtigt und die Zuordnung von Kunden zu bestimmten Arten von Touren bzw. Fahrzeugen anhand der Anforderung der Kunden durch die Vergabe von Constraints erlaubt. Eine wichtige, mit diesem Format eingeführte Neuerung ist die Möglichkeit zur Berücksichtigung der Bearbeitung der Transportfahrzeuge vor und nach einer Tour. Diese Möglichkeit ist nicht nur bei Problemstellungen aus dem Bereich von Werttransporten relevant, sondern grundsätzlich bei logistischen Systemen, bei denen die Be– und Entladung bzw. Auf– und Abrüstung von Transportfahrzeugen an Stützpunkten mit beschränkter Anzahl von Bearbeitungsplätzen stattfinden muss.

Bei der Arbeit am Prototypen sowie während der Tests stellte sich die Art der Benutzerführung bei der Interaktion als zur Problemstellung treffend heraus. Die implementierte Funktionalität zur Optimierung durch den Dispatcher beschränkt sich auf das Wesentliche, dafür wurde der Programmierung von Funktionalitäten zur Generierung eines der jeweiligen Situation passenden Feedbacks für den Dispatcher mehr Raum gegeben.

Der zur Programmierung des Prototypen benötigte Aufwand stellte sich als wesentlich umfangreicher heraus, als dies zu Beginn der Arbeit eingeschätzt wurde. Durch die Abbildung der üblichen Randbedingungen wie Zeitfenster und Kapazitätsbeschränkungen zusammen mit der Abbildung der beiden Schwerpunkte der Randbedingungen, der Schleusen–Problematik und den verschiedenen Kundenarten, wurde das System sehr komplex. Bei Abschluss der Programmierarbeiten umfasste der Code des Systems (ohne Code des als Basis

benutzten Frameworks) knapp 20000 Zeilen in ca. 100 Programmdateien. Der Nettogehalt an Code wird dabei auf ca. 15000 Lines of Code geschätzt.

Bei Abschluss der Programmierarbeiten zeigte der Prototyp noch einige Mängel und wies auch einige Fehler auf. Ein Hauptproblem dabei ist die Synchronisation der Pläne zwischen dem gegenwärtigen optimalen Plan des Optimierungsalgorithmus und dem Plan, der vom Dispatcher interaktiv verändert wird. Nicht alle der geplanten Funktionalitäten konnten implementiert werden. Die Hauptziele der Implementation konnten jedoch verwirklicht werden. Der Prototyp ist in der Lage, Touren und Pläne unter Berücksichtigung der Schleusen–Problematik und der verschiedenen Kundenarten zu erzeugen und zu optimieren. Die Optimierung durch einen Dispatcher und die Simulation des Planungshorizonts sind ebenfalls möglich. Der Prototyp stellt damit eine gute Basis zur Entwicklung von Systemen zur interaktiven und dynamischen Optimierung von Logistik–Problemen dar.

Erfreulich war der Umstand, dass sich das als Grundlage zur Entwicklung benutzte Framework des Autors SAF als stabile Basis bei der Arbeit erwies. Der Aufwand an Programmierarbeit konnte dadurch erheblich verringert werden. Auch die Integration der Funktionalität zur Interaktion des Dispatchers war relativ problemlos.

6.2 Ausblick

Im gegenwärtigen Stand der Arbeit erscheint die Lösung der Probleme bei der Synchronisation von Plänen als dringlichste Aufgabe. Daneben zeichnen sich mehrere Möglichkeiten zur Weiterentwicklung ab, von denen einige im Folgendenden genannt werden:

1. Entwicklung weiterer Modifikationsoperatoren für den Optimierungsalgorithmus. Der Schwerpunkt sollte dabei auf der Entwicklung schnell ablaufender Modifkationsoperatoren liegen. Bei komplexeren Modifikationsoperatoren wie die von PISINGER UND ROPKE eingeführten Operatoren *Related Removal* und *Regret Heuristics* [vgl. Pisinger und Ropke, 2007, S.11,13] besteht die Frage, wie diese möglichst effizient implementiert werden können.

2. Erweiterung der Funktionalität des Depot Models. Im gegenwärtigen Stand werden von den Fahrzeugen die ihnen zugewiesenen Schleusenzeiten nach Konstruktion der Tour (mit nachfolgender Anpassung der Schleusenzeiten) beibehalten. Die Möglichkeit zur nachträgliche Anpassung der Schleusenzeiten durch den Dispatcher würde in einigen Fällen die Möglichkeiten zur Optimierung vergrößern, da hierdurch der (potentielle) Lösungsraum vergrößert werden würde.

3. Entwicklung neuer Interaktonsmöglichkeiten für den Dispatcher. Bei Tests der Funktionalitäten für die Interaktion des Dispatchers wäre in einigen Fällen ein Operator hilfreich gewesen, mit dem eine Tour an einer Tour Connection aufgetrennt und in zwei neue Touren überführt werden kann. Eine Realisierung dieser Funktion erscheint jedoch aufwendig, da hierfür auch die Funktionalität des Depot Models erweitert werden müsste.

Literaturverzeichnis

[BGV:1997 1997] / Hauptverband der gewerblichen Berufgenossenschaften. URL http://www.arbeitssicherheit.de/arbeitssicherheit/html/modules/bgvc/bgv_c/c7.pdf, 1997 (BGV C7). – Unfallverhütungsvorschrift Wach– und Sicherungsdienste. URL zuletzt geprüft am 25.08.2008

[Anderson u. a. 2000] ANDERSON, David ; ANDERSON, Emily ; LESH, Neal ; MARKS, Joe ; MIRTICH, Brian ; RATAJCZAK, David ; RYALL, Kathy: Human-Guided Simple Search. In: *Proceedings of the Seventeenth National Conference on Artificial Intelligence and Twelfth Conference on Innovative Applications of Artificial Intelligence* ACM Special Interest Group on Computer-Human Interaction (Veranst.), AAAI Press / The MIT Press, 2000, S. 209–216. – URL http://www.merl.com/reports/docs/TR2000-16.pdf. – URL zuletzt geprüft am 25.08.2008. – ISBN 0-262-51112-6

[Attanasio u. a. 2007] ATTANASIO, Andrea ; BREGMAN, Jay ; GHIANI, Gianpaolo ; MANNI, Emanuele: *Operations Research | Computer Science Interfaces*. Bd. 38: *Real-Time Management at eCourier LTD*. Kap. 10, S. 219–238. In: *Dynamic Fleet Management* Bd. 38. New York, NY 10013, USA : Springer Science+Business Media, 2007. – ISBN 978-0-387-71721-0

[Bent und Hentenryck 2003] BENT, Russell ; HENTENRYCK, Pascal V.: Dynamic Vehicle Routing with Stochastic Requests. In: *Proceedings of the eighteenth international joint conference on artificial intelligence (IJCAI-2003)*. Oxford, UK : Elsevier Science Ltd., 2003, S. 1362–1363. – URL http://citeseer.ist.psu.edu/bent03dynamic.html. – URL zuletzt geprüft am 25.08.2008

[Bent und Hentenryck 2004] BENT, Russell ; HENTENRYCK, Pascal V.: Scenario-Based Planning for Partially Dynamic Vehicle Routing with Stochastic Customers. In: *Operations Research* 52 (2004), Nr. 6, S. 977–987. – ISSN 0030-364X

[Bent und Hentenryck 2006] BENT, Russell ; HENTENRYCK, Pascal V.: A two-stage hybrid algorithm for pickup and delivery vehicle routing problems with time windows. In: *Comput. Oper. Res.* 33 (2006), Nr. 4, S. 875–893. – URL http://www.cs.brown.edu/people/rbent/pickup.ps. – URL zuletzt geprüft am 25.08.2008. – ISSN 0305-0548

[Benyahia und Potvin 1995] BENYAHIA, Ilham ; POTVIN, Jean-Yves: Generalization and refinement of route construction heuristics using genetic algorithms. In: *IEEE International Conference on Evolutionary Computation 1995*, 1995, S. 39–43. – ISBN 0-7803-2759-4

[Boudali u. a. 2004] BOUDALI, Imen ; FKI, Wajdi ; GHEDIRA, Khaled: How to Deal with the VRPTW by using Multi-Agent Coalitions. In: *HIS '04: Proceedings of the Fourth International Conference on Hybrid Intelligent Systems*. Washington, DC, USA : IEEE Computer Society, 2004, S. 416–421. – ISBN 0-7695-2291-2

[Boudali u. a. 2005] BOUDALI, Imen ; FKI, Wajdi ; GHEDIRA, Khaled: An Interactive Distributed Approach For The Vrp With Time Windows. In: *International Journal of Simulation* 6 (2005), Nr. 10, S. 48–59

[Branke u. a. 2005] BRANKE, Jürgen ; MIDDENDORF, Martin ; NOETH, Guntram ; DESSOUKY, Maged: Waiting Strategies for Dynamic Vehicle Routing. In: *Transportation Science* 39 (2005), Nr. 3, S. 298–312. – URL http://pacosy.informatik.uni-leipzig.de/pv/Forschung/Papers/TransportationSci-Pap.pdf. – URL zuletzt geprüft am 25.08.2008. – ISSN 1526-5447

[Colorni u. a. 1991] COLORNI, Alberto ; DORIGO, Marco ; MANIEZZO, Vittorio: Distributed optimization by ant colonies. In: *Proceedings of the European Conference on Artificial Life*. Amsterdam : Elsevier, 1991, S. 143–142. – URL http://www.cs.ualberta.ca/~bulitko/F02/papers/IC.06-ECAL92.pdf. – URL zuletzt geprüft am 25.08.2008

[Coltorti und Rizzoli 2007] COLTORTI, Dario ; RIZZOLI, Andrea E.: Ant colony optimization for real-world vehicle routing problems. In: *SIGEVOlution* 2 (2007), Nr. 2, S. 2–9. – ISSN 1931-8499

[Cordeau 2006] CORDEAU, Jean-François: A branch-and-cut algorithm for the dial a ride problem. In: *Operations Research* (2006), Nr. 54, S. 573–586. – URL http://www.iro.umontreal.ca/~marcotte/PLU6000/PLU6000_H04/Cordeau1.pdf. – URL zuletzt geprüft am 25.08.2008

[Cordeau u. a. Due 2008] CORDEAU, Jean-François ; LAPORTE, Gilbert ; ROPKE, Stefan: *Operations Research | Computer Science Interfaces*. Bd. 43: *Recent Models and Algorithms for the One-to-One Pickup and Delivery Problems*. In: *The Vehicle Routing Problem* Bd. 43. New York, NY 10013, USA : Springer Science+Business Media, LLC, June Due 2008. – URL http://www.di.ku.dk/~sropke/Papers/Recent%20Models%20and%20Algorithms%20for%20One-to-One%20Pickup%20and%20Delivery%20Problems_revised.pdf. – URL zuletzt geprüft am 25.08.2008. – ISBN 978-0-387-77777-1

[Croes 1958] CROES, G.A.: A method for solving traveling salesman problems. In: *Operations Research* 6 (1958), Nr. 791–812

[Danzig und Ramser 1959] DANZIG, G.B. ; RAMSER, J.H.: The Truck Dispatching Problem. In: *Management Science* 6 (1959), Oct, Nr. 1, S. 80–91

[Dorigo und Stützle 2003] DORIGO, Marco ; STÜTZLE, Thomas: *The ant colony optimization metaheuristic: Algorithms, applications, and advances*. S. 251–285. In: GLOVER, Fred W. (Hrsg.) ; KOCHENBERGER, Gary A. (Hrsg.): *Handbook of metaheuristics*. 3300 AZ Dordrecht, The Netherlands : Springer Netherlands, 2003. – ISBN 1402072635

[Dror u. a. 1994] DROR, Moshe ; LAPORTE, Gilbert ; TRUDEAU, Pierre: Vehicle routing with split deliveries. In: *Discrete Appl. Math.* 50 (1994), Nr. 3, S. 239–254. – ISSN 0166-218X

[Duncan 1995] DUNCAN, Tim: Experiments in the use of neighbourhood search techniques for vehicle routing / University of Edinburgh, Artificial Intelligence Applications Institute. Edinburgh EH1 1HN, UK : University of Edinburgh, June 1995 (AIAI-TR-176). – Forschungsbericht. – URL http://neo.lcc.uma.es/radi-aeb/WebVRP/data/articles/duncan95experiments.pdf. URL zuletzt geprüft am 25.08.2008

[Flatberg u. a. 2005] FLATBERG, Truls ; HASLE, Geir ; KLOSTER, Oddvar ; NILSSEN, Eivind J. ; RIISE, Atle: Dynamic and Stochastic Aspects in Vehicle Routing - A Literature Survey / SINTEF. 0314 Oslo, Norway : SINTEF, June 2005 (STF90A05413). – Forschungsbericht. – URL http://www.doit.sintef.no/files/Dynamic%20ans%20Stochastic%20Aspects%20in%20Vehicle%20Routing%20-%20A%20Literature%20Survey.pdf. URL zuletzt geprüft am 25.08.2008

[Flatberg u. a. 2007] FLATBERG, Truls ; HASLE, Geir ; KLOSTER, Oddvar ; NILSSEN, Eivind J. ; RIISE, Atle: *Operations Research | Computer Science Interfaces*. Bd. 38: *Dynamic and stochastic vehicle routing in practice*. Kap. 3, S. 41–64. In: *Dynamic Fleet Management* Bd. 38. New York, NY 10013, USA : Springer Science+Business Media, 2007. – ISBN 978-0-387-71721-0

[Flood 1956] FLOOD, M.M.: The traveling–salesman problem. In: *Operations Research* 4 (1956), S. 61–75

[Gambardella u. a. 1999] GAMBARDELLA, Luca M. ; TAILLARD Éric ; AGAZZI, Giovanni: *MACS-VRPTW: a multiple ant colony system for vehicle routing problems with time windows*. S. 63–76. In: *New ideas in optimization*. Maidenhead, UK, England : McGraw-Hill Ltd., UK, 1999. – URL http://www.idsia.ch/%7Emonaldo/VRP/tr-idsia-06-99.pdf. – URL zuletzt geprüft am 25.08.2008. – ISBN 0-07-709506-5

[Gamma u. a. 1994] GAMMA, Erich ; HELM, Richard ; JOHNSON, Ralph ; VLISSIDES, John: *Design Patterns*. Addison-Wesley, 1994. – ISBN 0-201-63361-2

[Gendreau u. a. 1999] GENDREAU, Michel ; GUERTIN, Francois ; POTVIN, Jean-Yves ; TAILLARD, Eric: Parallel Tabu Search for Real-Time Vehicle Routing and Dispatching. In: *Transportation Science* 33 (1999), Nr. 4, S. 381–390. – URL http://mistic.heig-vd.ch/taillard/articles.dir/GGPT.pdf. – URL zuletzt geprüft am 25.08.2008. – ISSN 1526-5447

[Gendreau u. a. 2001] GENDREAU, Michel ; LAPORTE, Gilbert ; POTVIN, Jean-Yves: *Metaheuristics for the Vehicle Routing Problem*. S. 129–154. In: *The Vehicle Routing Problem*. Philadelphia, PA 19104-2688 USA : Society for Industrial and Applied Mathematics, 2001 (SIAM Monographs on Discrete Mathematics and Applications). – URL http://neo.lcc.uma.es/radi-aeb/WebVRP/data/articles/G9852R3.ps. – URL zuletzt geprüft am 25.08.2008. – ISBN 0-89871-579-2

[Hall und Partyka 2008] HALL, Randolph W. ; PARTYKA, Janice G.: On the Road to Mobility. In: *OR/MS Today* 35 (2008), Nr. 1, S. 44–47. – URL zuletzt geprüft am 25.08.2008

[Held 2008] HELD, Rudolf: Entwicklung eines Frameworks für stochastische Algorithmen / Hochschule für angewandte Wissenschaften Hamburg, Fakultät Technik und Informtik, Department Informatik. HAW Hamburg, Berliner Tor 2, 20099 Hamburg, Germany, Jan 2008. – Studienarbeit. – URL http://www.knusperfisch.de/ki/papers/saf.1.02.pdf. URL zuletzt geprüft am 25.08.2008

[Holland 1975] HOLLAND, John H.: *Adaptation in natural and artificial systems*. University of Michigan Press, 1975

[Ichoua u. a. 2007] ICHOUA, Soumia ; GENDREAU, Michael ; POTVIN, Jean-Yves: *Operations Research | Computer Science Interfaces*. Bd. 38: *Planned route optimization for real–time vehicle routing*. Kap. 1, S. 1–18. In: *Dynamic Fleet Management* Bd. 38. New York, NY 10013, USA : Springer Science+Business Media, 2007. – ISBN 978-0-387-71721-0

[Ichoua u. a. 2000] ICHOUA, Soumia ; GENDREAU, Michel ; POTVIN, Jean-Yves: Diversion Issues in Real-Time Vehicle Dispatching. In: *Transportation Science* 34 (2000), Nr. 4, S. 426–438. – URL http://www.iro.umontreal.ca/~potvin/soumia.pdf. – URL zuletzt geprüft am 25.08.2008. – ISSN 1526-5447

[Ickert u. a. 2007] ICKERT, Lutz ; MATTHES, Ulricke ; ROMMERSKIRCHEN, Stefan ; WEYAND, Emily ; SCHLESINGER, Michael ; LIMBERS, Jan: Abschätzung der langfristigen Entwicklung des Güterverkehrs in Deutschland bis 2050 (Schlussbericht) / ProgTrans AG im Auftrag des Bundesministeriums für Verkehr, Bau und Stadtentwicklung. CH-4001 Basel : ProgTrans, Mai 2007 (Projekt-Nr. 26.0185/2006). – Forschungsbericht. – URL http://www.bmvbs.de/Anlage/original_999441/Gueterverkehrs-prognose-2050.pdf. URL zuletzt geprüft am 25.08.2008

[Johnson und McGeoch 1997] JOHNSON, David S. ; MCGEOCH, Lyle.A.: *The traveling salesman problem: a case study*. S. 215–310. In: *Local Search in Combinatorial Optimization*. Chichester, UK : Wiley, 1997. – URL www.research.att.com/~dsj/papers/TSPchapter.ps. – URL zuletzt geprüft am 25.08.2008. – ISBN 0471948225

[Kennedy und Eberhart 2001] KENNEDY, James ; EBERHART, Russel C. ; FOGEL, David B. (Hrsg.): *Swarm Intelligence*. San Francisco, CA 94104-3205, USA : Morgan Kaufmann Publishers, 2001 (The Morgan Kaufmann Series in Evolutionary Computing). – ISBN 1-55860-595-9

[Kilby u. a. 1998] KILBY, Philip ; PROSSER, Patrick ; SHAW, Paul: Dynamic VRPs: A Study of Scenarios / University of Strathclyde. publisher unknown, 1998 (APES-06-1998). – Forschungsbericht. – URL users.rsise.anu.edu.au/~pjk/papers/Kilb98Dynamic.ps. URL zuletzt geprüft am 25.08.2008

[Kindervater und Savelsbergh 1997] KINDERVATER, Gerard A. ; SAVELSBERGH, Martin W.: *Vehicle Routing: Handling Edge Exchanges*. S. 337–360. In: *Local Search in Combinatorial Optimization*. Chichester, UK : Wiley, 1997. – URL http://www2.isye.gatech.edu/people/faculty/Martin_Savelsbergh/publications/lsco.pdf. – URL zuletzt geprüft am 25.08.2008. – ISBN 0471948225

[Kirkpatrick u. a. 1983] KIRKPATRICK, S. ; GELATT, C. D. ; VECCHI, M. P.: Optimization by Simulated Annealing. In: *Science* 220 (1983), May, Nr. 4598, S. 671 – 680. – URL www.fisica.uniud.it/~{}ercolessi/MC/kgv1983.pdf. – URL zuletzt geprüft am 25.08.2008

[Klau u. a. 2002] KLAU, Gunnar W. ; LESH, Neal ; MARKS, Joe ; MITZENBACHER, Michael ; SCHÄFER, Guy T.: The HuGS Platform: A Toolkit for Interactive Optimization / Mitsubishi Electric Research Laboratories. Cambridge, MA 02139, USA : Mitsubishi Electric Research Laboratories, Dezember 2002 (TR2002-08). – Mitsubishi Electric Research Laboratories. – URL http://www.merl.com/reports/docs/TR2002-08.pdf. URL zuletzt geprüft am 25.08.2008

[Klauck und Maas 1999] KLAUCK, Christoph ; MAAS, Christoph: *Inf & Ing, Vorlesungen zum Informatik- und Ingenieurstudium*. Bd. 1: *Graphentheorie und Operations Research für Studierende der Informatik*. 3. Augsburg : Wißner, 1999

[Kohout und Erol 1999] KOHOUT, Robert ; EROL, Kutluhan: In-time agent-based vehicle routing with a stochastic improvement heuristic. In: *AAAI '99/IAAI '99: Proceedings of the sixteenth national conference on Artificial intelligence and the eleventh Innovative applications of artificial intelligence conference innovative applications of artificial intelligence*. Menlo Park, CA, USA : American Association for Artificial Intelligence, 1999, S. 864–869. – ISBN 0-262-51106-1

[Kopfer und Schonberger 2002] KOPFER, Herbert ; SCHONBERGER, Jörn: Interactive Solving of Vehicle Routing and Scheduling Problems: Basic Concepts and Qualification of Tabu Search Approaches. In: *35th Annual Hawaii International Conference on System Sciences (HICSS02)* Bd. 3. Los Alamitos, CA 90720, USA : IEEE Computer Society, Januar 2002, S. 1425–1434. – URL http://csdl2.computer.org/comp/proceedings/hicss/2002/1435/03/14350084.pdf. – URL zuletzt geprüft am 25.08.2008. – ISBN 0-7695-1435-9

[Kumar 1992] KUMAR, V.: *Search, Branch–and–Bound*. S. 1468–1472. In: *Encyclopedia of Artificial Intelligence*. New York, NY, USA : John Wiley & Sons, Inc., 1992. – ISBN 0471503053

[Laporte u. a. 2000] LAPORTE, Gilbert ; GENDREAU, Michel ; POTVIN, Jean-Yves ; SEMET, Frédéric: Classical and modern heuristics for the vehicle routing problem. In: *International Transactions in Operational Research* 7 (2000), Nr. 4-5, S. 285–300. – URL http://neo.lcc.uma.es/radi-aeb/WebVRP/data/articles/G9921R.ps. – URL zuletzt geprüft am 25.08.2008

[Laporte und Semet 2001] LAPORTE, Gilbert ; SEMET, Frédéric: *Classical Heuristics for the Vehicle Routing Problem*. S. 109–128. In: *The Vehicle Routing Problem*. Philadelphia, PA 19104-2688, USA : Society for Industrial and Applied Mathematics, 2001 (SIAM Monographs on Discrete Mathematics and Applications). – URL http://neo.lcc.uma.es/radi-aeb/WebVRP/data/articles/G9854R2.ps. – URL zuletzt geprüft am 25.08.2008. – ISBN 0-89871-579-2

[Larsen u. a. 2007] LARSEN, Allan ; MADSEN, Olli B. G. ; SOLOMON, Marius M.: *Operations Research | Computer Science Interfaces*. Bd. 38: *Classification of dynamic vehicle routing systems*. Kap. 2, S. 19–40. In: *Dynamic Fleet Management* Bd. 38. New York, NY 10013, USA : Springer Science+Business Media, 2007. – ISBN 978-0-387-71721-0

[Li und Lim 2003] LI, Haibing ; LIM, Andrew: A Metaheuristic for the Pickup and Delivery Problem with Time Windows. In: *International Journal on Artificial Intelligence Tools* 12 (2003), Nr. 2, S. 173–186

[Lin 1965] LIN, S.: Computer Solutions of the traveling salesman problem. In: *Bell Syst. Tech. J.* (1965), Nr. 44, S. 2245–2269

[Metropolis u. a. 1953] METROPOLIS, N. ; ROSENBLUTH, A. W. ; ROSENBLUTH, M. N. ; TELLER, A. H. ; TELLER, E.: Equation of State Calculations by Fast Computing Machines. In: *The Journal of Chemical Physics* 21 (1953), June, Nr. 6, S. 1087–1092. – URL http://people.scs.fsu.edu/~{}beerli/mcmc/metropolis-et-al-1953.pdf. – URL zuletzt geprüft am 25.8.2008

[Montemanni u. a. 2002] MONTEMANNI, Roberto ; GAMBARDELLA, Luca M. ; RIZZOLI, Andrea E. ; DONATI, Alberto V.: A new algorithm for a Dynamic Vehicle Routing Problem based on Ant Colony System. CH–6928 Manno, Switzerland : Istituto Dalle Molle Di Studi Sull Intelligenza Artificiale, 2002 (23-02). – Forschungsbericht. – URL ftp://ftp.idsia.ch/pub/techrep/IDSIA-23-02.pdf.gz. URL zuletzt geprüft am 25.08.2008

[Ombuki u. a. 2006] OMBUKI, Beatrice ; ROSS, Brian J. ; HANSHAR, Franklin: Multi-Objective Genetic Algorithms for Vehicle Routing Problem with Time Windows. In: *Applied Intelligence* 24 (2006), Nr. 1, S. 17–30. – URL http://www.lania.mx/~ccoello/EMOO/ombuki06.pdf.gz. – URL zuletzt geprüft am 25.08.2008. – ISSN 0924-669X

[Or 1976] OR, I.: *Traveling Salesman-Type Combinatorial Problmes and their Relation to the Logistics of Blood Banking*. Evanston, Northwestern University, Dept. of Industrial Engineering and Managment Sciences, Dissertation, 1976

[Osman 1993] OSMAN, Ibrahim H.: Metastrategy simulated annealing and tabu search algorithms for the vehicle routing problem. In: *Annals of Operations Research* 41 (1993), Nr. 4, S. 421–451. – ISSN 0254-5330

[Paulick 2007] PAULICK, Andreas: Neue Sicherheitsvorschriften der BDGW / Bundesvereinigung deutscher Geld– und Wertdienste. URL http://www.bdws.de/cms/DSD/1-03/07.pdf, 2007 (BGV C7). – Unfallverhütungsvorschrift Wach– und Sicherungsdienste. URL zuletzt geprüft am 25.08.2008

[Pisinger und Ropke 2007] PISINGER, David ; ROPKE, Stefan: A general heuristic for vehicle routing problems. In: *Computers and Operations Research* 34 (2007), August, Nr. 8, S. 2403–2435. – URL http://www.diku.dk/~sropke/Papers/GeneralVRP_TechRep.pdf. – URL zuletzt geprüsft am 25.08.2008. – ISSN 0305-0548

[Potvin und Bengio 1996] POTVIN, Jean-Yves ; BENGIO, Samy: The vehicle routing problem with time windows – Part II: Genetic Search. In: *Journal on Computing* 8 (1996), S. 1996. – URL http://citeseer.ist.psu.edu/509992.html. – URL zuletzt geprüft am 25.08.2008

[Robusté u. a. 1990] ROBUSTÉ, Francesco ; DAGANZO, Carlos F. ; SOULEYRETTE, Reginald R.: Implementing Vehicle Routing Models. In: *Transportation Research* 24B (1990), Nr. 4, S. 263–286. – URL http://www.sciencedirect.com/science?_ob=MImg&_imagekey=B6V99-466CD71-3Y-1&_cdi=5893&_orig=browse&_coverDate=08/31/1990&_sk=999759995&view=c&wchp=dGLbVtz-zSkWz&_acct=C000000152&_version=1&_userid=4420&md5=81fbebbf1849baab427c1e15a5e4b5c5&ie=f.pdf. – URL zuletzt geprüft am 25.08.2008

[Ropke und Cordeau 2007] ROPKE, Stefan ; CORDEAU, Jean-François: Branch-and-Cut-and-Price for the Pickup and Delivery Problem with Time Windows / Canada Research Chair in Logistics and Transportation. Montréal, H3T 2A7 Canada : Canada Research Chair in Logistics and Transportation, HEC, November 2007 (C7PQMR PO2006-21-X). – Forschungsbericht. – URL http://www.diku.dk/~sropke/Papers/Branch-and-Cut-and-Price%20for%20the%20Pickup%20and%20Delivery%20Problem%20with%20Time%20Windows.pdf. URL zuletzt geprüft am 25.08.2008

[Ropke u. a. 2007] ROPKE, Stefan ; CORDEAU, Jean-François ; LAPORTE, Gilbert: Models and a Branch-and-Cut Algorithm for Pickup and Delivery Problems with Time Windows. In: *Networks* 49 (2007), Nr. 4, S. 258–272. – URL http://www.di.ku.dk/~sropke/Papers/BAC-pdptw-revised.pdf. – URL zuletzt geprüft am 25.08.2008

[Ropke und Pisinger 2006a] ROPKE, Stefan ; PISINGER, David: An Adaptive Large Neighborhood Search Heuristic for the Pickup and Delivery Problem with Time Windows. In: *Transportation Science* 40 (2006), November, Nr. 4, S. 455–472. – URL http://www.di.ku.dk/~sropke/Papers/PDPTW_techRep.pdf. – URL zuletzt geprüft am 25.08.2008. – ISSN 1526-5447

[Ropke und Pisinger 2006b] ROPKE, Stefan ; PISINGER, David: A unified heuristic for a large class of Vehicle Routing Problems with Backhauls. In: *European Journal of Operational Research* 127 (2006), June, Nr. 3, S. 750–775. – URL http://www.di.ku.dk/~sropke/Papers/VRPB-heur.pdf. – URL zuletzt geprüft am 25.08.2008

[Savelsbergh 1985] SAVELSBERGH, Martin W. P.: Local search for routing problems with time windows. In: *Annals of Operations Research* 4 (1985), Dezember, Nr. 1, S. 285–305. – URL http://www.springerlink.com/content/m0431120n4851m82/fulltext.pdf. – URL zuletzt geprüft am 25.08.2008. – ISSN 0254-5330

[Savelsbergh und Sol 1995] SAVELSBERGH, Martin W. P. ; SOL, M.: The General Pickup and Delivery Problem. In: *Transportation Science* (1995), Nr. 29, S. 17–29. – URL http://www2.isye.gatech.edu/people/faculty/Martin_Savelsbergh/publications/ts29.pdf. – URL zuletzt geprüft am 25.08.2008

[Schafhausen u. a. 2005] SCHAFHAUSEN, Franzjosef ; HARNISCH, Astrid ; (REDAKTION), Katrin A.: *Nationales Klimaschutzprogram 2005, sechster Bericht der Interministeriellen Arbeitsgruppe CO_2–Reduktion*. Referat Öffentlichkeitsarbeit, 11055 Berlin : Bundesministerium für Umwelt, Naturschutz und Reaktorsicherheit (BMU), August 2005. – URL http://www.bmu.de/files/klimaschutz/downloads/application/pdf/klimaschutzprogramm_2005_lang.pdf. – URL zuletzt geprüft am 25.08.2008

[Scott u. a. 2002] SCOTT, Stacey D. ; LESH, Neal ; KLAU, Gunnar W.: Investigating human-computer optimization. In: *CHI '02: Proceedings of the SIGCHI conference on Human factors in computing systems*. New York, NY, USA : ACM, 2002, S. 155–162. – URL http://www.merl.com/reports/docs/TR2001-39.pdf. – URL zuletzt geprüft am 25.08.2008. – ISBN 1-58113-453-3

[Shaw 1997] SHAW, Paul: A new local search algorithm providing high quality solutions to vehicle routing problems / APES Group, Department of Computer Science, University of Strathclyde. publisher unknown, July 1997. – technical report. – URL http://citeseer.ist.psu.edu/shaw97new.html. URL zuletzt geprüft am 25.08.2008

[Shaw 1998] SHAW, Paul: *Lecture Notes in Computer Science*. Bd. 1520/1998: *Using Constraint Programming and Local Search Methods to Solve Vehicle Routing Problems*. S. 417–431. In: *Principles and Practice of Constraint Programming* Bd. 1520/1998. New York, NY 10013, USA : Springer Science+business Media, LLC, 1998. – URL http://citeseer.ist.psu.edu/83532.html. – URL zuletzt geprüft am 25.08.2008. – ISBN 978-3-540-65224-3

[Taillard 1993] TAILLARD, Eric: Parallel Iterative Search Methods for Vehicle Routing Problems. In: *Networks* (1993), Nr. 23, S. 661–676. – URL http://www3.interscience.wiley.com/cgi-bin/fulltext/118639195/PDFSTART. – URL zuletzt geprüft am 25.08.2008

[Tangiah 1993] TANGIAH, Sam R.: Vehicle Routing with Time Windows using Genetic Algorithms / Slippery Rock University. Slippery Rock, PA, USA, 1993 (SRU–CpSc–TR–93–23). – Forschungsbericht. – URL http://citeseer.ist.psu.edu/481594.html. URL zuletzt geprüft am 25.08.2008

[Wah u. a. 1995] WAH, Benjamin W. ; IEUMWANANONTHACHAI, Arthur ; CHU, Lon-Chan ; AIZAWA, Akiko N.: Genetics-Based Learning of New Heuristics: Rational Scheduling of Experiments and Generalization. In: *IEEE Transactions on Knowledge and Data Engineering* 7 (1995), October, Nr. 5, S. 763–785. – URL http://ieeexplore.ieee.org/iel3/69/9912/00469821.pdf. – URL zuletzt geprüft am 25.08.2008

[Whitley 1994] WHITLEY, Darrell: A Genetic Algorithm Tutorial. In: *Statistics and Computing* 4 (1994), S. 65–85. – URL http://www.cs.colostate.edu/~genitor/MiscPubs/tutorial.pdf. – URL zuletzt geprüft am 25.08.2008

[Zeimpekis u. a. 2007] ZEIMPEKIS, V. ; MINIS, I. ; MAMASSIS, K. ; GIAGLIS, G.M.: *Operations Research | Computer Science Interfaces*. Bd. 38: *Dynamic Management of a delayed Delivery Vehicle in a City Logistics Environment*. Kap. 9, S. 197–217. In: *Dynamic Fleet Management* Bd. 38. New York, NY 10013, USA : Springer Science+Business Media, 2007. – ISBN 978-0-387-71721-0

Versicherung über Selbstständigkeit

Hiermit versichere ich, dass ich die vorliegende Arbeit im Sinne der Prüfungsordnung nach §24(5) ohne fremde Hilfe selbstständig verfasst und nur die angegebenen Hilfsmittel benutzt habe.

Hamburg, 28. August 2008	
Ort, Datum	Unterschrift