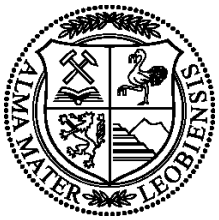


Masterarbeit

Entwicklung einer automatischen Tourenzusammenstellung

Das Vehicle Routing Problem in der Praxis

eingereicht an der

Montanuniversität Leoben

erstellt am

Lehrstuhl für Angewandte Mathematik

Vorgelegt von:
Georg WALCHSHOFER, BSc.
0535025

Betreuer/Gutachter:
Ao. Univ. Prof.Dipl.-Ing. Dr. tech. Norbert Seifter

Graz, 14.09.2012

Eidesstattliche Erklärung

Ich erkläre an Eides statt, dass ich diese Arbeit selbständig verfasst, andere als die angegebenen Quellen und Hilfsmittel nicht benutzt und mich auch sonst keiner unerlaubten Hilfsmittel bedient habe.

(Georg Walchshofer)

Graz, 14.09.2012

Danksagung

An dieser Stelle möchte ich all jenen einen Dank aussprechen, die mich während meines gesamten Studiums, und insbesondere bei der Erstellung dieser Arbeit, unterstützten. Dabei gilt das größte Dankeschön meiner lieben Lebensgefährtin Julia Kraber, die nicht nur immer großes Verständnis dafür aufbrachte, dass die gemeinsame Freizeitgestaltung des Öfteren dem Vorantreiben der Arbeit zum Opfer gefallen ist, sondern mir auch immer neue Motivation gab, wenn der eigene Antrieb etwas nachließ. Auch dass sie sich immer so liebevoll um unsere gemeinsamen Kinder Lukas und Marie kümmerte, half, mich voll und ganz auf die Arbeit konzentrieren zu können. Des Weiteren hörte sie immer geduldig zu, wenn ich meine Gedanken in Form eines Gesprächs sortieren wollte, und das, obwohl das in dieser Arbeit behandelte Thema sicherlich nicht zu ihren Lieblingsthemen gehört.

Ebenfalls ein großes Dankeschön gilt meinen Eltern, die mich während meines gesamten Studiums, vor allem auch finanziell, unterstützten. Weiters möchte ich auch dem Betreuer dieser Arbeit, Ao. Univ. Prof. Dipl.-Ing. Dr. tech. Norbert Seifter, meinen Dank aussprechen, da mich dieser immer in meiner Vorgehensweise bekräftigte, und mir dabei mit hilfreichen Tipps Unterstützung bat. Abschließend möchte ich mich auch bei dem Unternehmen Elefant Holding AG bedanken, das mir von Beginn an das Vertrauen schenkte, und mir somit die Möglichkeit gab, meine eigenen Ideen zu verfolgen und letztlich auch umzusetzen.

Kurzfassung

Das Vehicle Routing Problem ist eines der bekanntesten kombinatorischen Optimierungsprobleme, mit denen Unternehmen im alltäglichen Geschäftsprozess konfrontiert werden. Auch wenn dabei das Kernproblem, das Aufteilen von vorhandenen oder geplanten Aufträgen auf verschiedene Touren zu möglichst geringen Gesamtkosten, immer dasselbe ist, können sich die konkreten Ausprägungen dieses Problems von Unternehmen zu Unternehmen sehr stark unterscheiden. Diese Unterschiede zeigen sich insbesondere in Form von verschiedenen Neben- und Randbedingungen, sowie unterschiedlichen, bzw. unterschiedlich gewichteten, Zielsetzungen. Aus diesem Grund wird für ein effizientes Lösen dieses Problems in vielen Fällen ein individuell angefertigtes bzw. adaptiertes Programm benötigt, um den speziellen Anforderungen des Unternehmens bestmöglich gerecht zu werden.

Im Zuge dieser Arbeit wird solch eine angepasste Lösung (bzw. ein solches Programm) für die Elefant Holding AG, einem international produzierenden Möbelkonzern, entwickelt. Dabei werden zunächst das zu Grunde liegende Problem und dessen Erweiterungen erörtert, sowie die wichtigsten Ansätze zur Lösung dieser Probleme gezeigt. Aufbauend auf diesen Ansätzen werden für die konkret vorliegende Problemstellung verschiedene Verfahren zur automatischen Tourenzusammenstellung entwickelt, getestet und verglichen. Auf die mit der Neustrukturierung des gesamten Tourenplanungsprozesses einhergehenden organisatorischen Veränderungen, sowie auf das Programm, in das die Möglichkeit der automatischen Tourenzusammenstellung integriert wird, wird ebenfalls kurz eingegangen.

Abstract

The Vehicle Routing Problem is one of the most familiar problems in combinatorial optimization, which are an important part of the daily business in many companies. The central problem, the cost-effective allocation of orders to different tours, is always the same, but the concrete characteristic of this problem can vary widely. The main reasons for this variation are different constraints and different objectives. Because of this wide variation, companies often need an individual solution (and an individual program) to solve their Vehicle Routing Problem efficiently and to meet all their requirements.

In this thesis such an individual solution (and program) will be developed for the Elefant Holding AG, an international furniture producing company. First the underlying combinatorial optimization problem and its basic extensions and variants will be discussed, and the most important solution approaches are shown. Referring to these approaches, different heuristics for the concrete distribution problem of this company will be developed, analysed and compared. The organisational changes of the restructuring of the distribution-process and the program, in which the developed automatic allocation of orders is integrated, will also be presented.

Inhaltsverzeichnis

Abbildungsverzeichnis

1 Einleitung

Die Transportkosten machen, unter anderem ausgelöst durch die immer teurer werdenden Treibstoffe (und den Verschärfungen der Umweltbestimmungen), einen immer größeren Anteil an den Gesamtkosten eines (produzierenden) Unternehmens aus. Aus diesem Grund kann heute eine effizient gesteuerte Distribution entscheidende Kostenvorteile für ein Unternehmen bringen. Deshalb hat sich die Elefant Holding AG, die aus vier möbelproduzierenden Unternehmen (eines in Österreich, zwei in Ungarn und eines in Rumänien) besteht, dazu entschlossen, deren Tourenplanung einer umfangreichen Neustrukturierung zu unterwerfen. Die Erstellung des Konzepts, sowie dessen Umsetzung, erfolgt im Rahmen dieser Arbeit, wobei die organisatorischen Veränderungen hier nur am Rande erwähnt werden. Der Schwerpunkt dieser Arbeit liegt in der Erarbeitung einer Möglichkeit zur automatischen Tourenzusammenstellung, die in das, ebenfalls im Zuge dieser Arbeit entwickelte, Programm integriert werden soll. Durch dieses Programm, und insbesondere durch die Möglichkeit einer automatischen Tourenzusammenstellung, sollen sowohl der administrative Aufwand für die Tourenplanung, als auch die Transportkosten (durch effizientere Tourenzusammenstellungen), gesenkt werden.

Da der Schwerpunkt auf dieser automatischen Tourenzusammenstellung liegt, wird zunächst auf das zu Grunde liegende kombinatorische Optimierungsproblem – das Vehicle Routing Problem – näher eingegangen. Anschließend werden bereits bekannte Verfahren zur Lösung solcher Probleme gezeigt. Erst dann folgt jener Teil, in dem die Gegebenheiten innerhalb der Tourenplanung des erwähnten Unternehmens analysiert, und die daraus resultierenden Zielsetzungen, aufgezeigt werden. In diesem Teil wird auch auf das daraus abgeleitete Gesamtkonzept kurz eingegangen. Danach wird die Ausarbeitung der automatischen Tourenzusammenstellung erörtert, wobei nicht nur eine Beschreibung der verschiedenen, entwickelten Verfahren, sondern auch eine ausführliche Analyse der Ergebnisse, gezeigt wird. Den Abschluss dieser Arbeit macht ein Resümee über den Nutzen, der aus dem umgesetzten Konzept, insbesondere der entwickelten Möglichkeit zur automatischen Tourenzusammenstellung, gezogen werden kann.

2 Das Vehicle Routing Problem (VRP)

Jeden Tag beschäftigen sich unzählige Unternehmen auf der ganzen Welt – je nach Branche und Größe mehr oder weniger intensiv – mit einer Aufgabe die allgemein als Tourenplanung bezeichnet wird. Hinter dieser Aufgabe, dem Zuteilen von Aufträgen zu Fahrzeugen und in weiterer Folge zu Touren, steckt ein kombinatorisches Optimierungsproblem, heute bekannt als Vehicle Routing Problem (VRP), zu dem 1959 von Dantzig und Ramser erstmals ein Artikel in „Management Science" veröffentlicht wurde [8]. In diesem Artikel beschäftigen sich die Autoren mit der Auslieferung von Treibstoff eines zentralen Distributors zu mehreren Service-Stationen (Tankstellen), wobei sie dieses Problem als Truck Dispatching Problem bezeichnen und als Verallgemeinerung des Travelling-Salesman-Problem (TSP) beschreiben.

In den letzten Jahrzehnten wurde dem Vehicle Routing Problem große Aufmerksamkeit geschenkt und es wurde eine Vielzahl von Lösungsmöglichkeiten zu verschiedensten Ausprägungen des VRP publiziert. Diese große Aufmerksamkeit verdankt das VRP sowohl der ökonomischen Relevanz dieser Problemstellung, als auch der Faszination der enormen Komplexität dieses Problems. Selbst mit den effizientesten Algorithmen können lediglich Problemstellungen mit etwa 50 Kunden deterministisch gelöst werden ([42], Preface).

In diesem Kapitel werden das Vehicle Routing Problem und einige Ausprägungen näher erörtert. Dafür sollen zunächst die für das VRP relevanten Grundlagen der Graphentheorie sowie das Traveling-Salesman Problem (TSP) betrachtet werden, da dieses erstens als Vorstufe zu dem VRP gesehen werden kann und zweitens ein Teilproblem des letztgenannten darstellt. Nachdem anschließend die Ausgangsproblemstellung des VRP, das Capacitated Vehicle Routing Problem (CVRP), beschrieben und eine mathematische Formulierung des Problems gezeigt wird, wird auf die verschiedenen Möglichkeiten der Abwandlungen und Erweiterungen des VRP eingegangen.

2.1 Grundlagen und das TSP

Das Travelling-Salesman-Problem, auch bekannt als das Problem des Handlungsreisenden, zählt zu den kombinatorischen Optimierungsproblemen und ist eine der am meisten untersuchten Problemstellungen der kombinatorischen Optimierung. Diese Problemstellung wird in [9] auf S.95 in etwa wie folgt beschrieben: Ein Geschäftsmann muss in verschiedenen Städ-

ten Kunden besuchen und anschließend wieder zu seinem Heimatort zurückkehren. In welcher Reihenfolge soll der Geschäftsmann die einzelnen Kunden besuchen, damit der insgesamt zurückgelegte Weg beziehungsweise die dafür benötigte Zeit oder die dadurch entstehenden Kosten minimal sind?

Da für die Beschreibung dieses Problems und in weiterer Folge auch für die Beschreibung des VRP Graphen benötigt werden, werden an dieser Stelle die für diese Problemstellungen relevanten Definitionen und Notationen erörtert ([26], S.15ff):

Ein **Graph** $G = (V, E)$ besteht aus **Knoten** und **Kanten**, wobei die Menge $V = \{v_1, \dots, v_n\}$ alle Knoten und die Menge $E = \{(v_i, v_j) | v_i, v_j \in V(G), i \neq j\}$ alle Kanten beinhaltet. Eine Kante $e = (v, w)$ verbindet die zwei Knoten v und w. Können alle Kanten eines Graphen in beide Richtungen durchlaufen werden, d.h. sowohl von v nach w als auch von w nach v, wird von einem **ungerichteten Graphen** gesprochen, andernfalls handelt es sich um einen **gerichteten Graphen bzw. Digraphen**. Ein Digraph liegt in der Praxis zum Beispiel durch Einbahnsysteme im Straßennetz vor. Eine Spezialform eines ungerichteten Graphen ist der **vollständige Graph** bei dem alle Knoten direkt miteinander verbunden sind, d.h. für alle Knotenpaare $(v_i, v_j),\ v_i, v_j \in V,\ i \neq j$ gibt es eine verbindende Kante. Der Graph $H = \big(V(H), E(H)\big)$ ist ein **Teilgraph** von $G = \big(V(G), E(G)\big)$ wenn $V(H) \subseteq V(G)$ und$E(H) \subseteq E(G)$. Gilt $V(H) = V(G)$ ist der Teilgraph H **spannend**.

Eine **Kantenfolge** W in G ist eine Folge von Knoten und Kanten $v_1, e_1, v_2, \dots, v_k, e_k, v_{k+1}$ wobei $e_i = (v_i, v_{i+1}) \in E(G)$ für$i = 1, \dots, k$. Ist bei solch einer Kantenfolge die Bedingung $e_i \neq e_j$ für alle $1 \leq i < j \leq k$ erfüllt (also kommt eine Kante höchstens einmal vor) so spricht man von einem **Weg**. Gilt zusätzlich noch $v_1 = v_{k+1}$ so ist dieser Weg **geschlossen**. Hat ein Graph $P = (\{v_1, \dots, v_{k+1}\}, \{e_1, \dots, e_k\})$ die Eigenschaften dass $v_i \neq v_j$ für $1 \leq i < j \leq k + 1$ und ist die Folge $v_1, e_1, v_2, \dots, v_k, e_k, v_{k+1}$ ein Weg, so bezeichnet man diesen Weg als **Pfad**. Ist der Weg geschlossen so spricht man von einem **Kreis**. Ist dieser Graph P nun ein Teilgraph von G so spricht man von einem Pfad bzw. Kreis in G, ist P spannend, so liegt ein **hamiltonscher Weg** bzw. ein **hamiltonscher Kreis** vor.

Das Traveling-Salesman Problem kann nun wie folgt beschrieben werden (siehe [26], S.416): Gegeben ist ein Graph $G = (V, E)$ mit n Knoten ($n - 1$ Kunden und der Startpunkt des Handelsreisenden) wobei jeder Kante $e \in E$ Kosten bzw. eine Gewichtung $c(e)$ (in der Regel die Entfernung) zugeordnet ist. Dadurch erhält man einen **bewerteten Graphen** $G = (V, E, C(E))$, wobei $C(E)$ die Kosten für die einzelnen Kanten enthält. Gesucht ist ein hamiltonscher Kreis mit minimalen Kosten. Ist der zugrundeliegende Graph gerichtet, so

spricht man von einem **asymmetrischen** TSP, ist er ungerichtet von einem **symmetrischen**.

Eine zweite Möglichkeit, das Traveling-Salesman Problem zu beschreiben, ist die mathematische Formulierung als ganzzahliges lineares Optimierungsproblem. Diese Art der Formulierung ist, wie später gezeigt wird, für die Entwicklung von Lösungsverfahren von Bedeutung. Die hier gezeigte Formulierung orientiert sich im Wesentlichen an den Ausführungen in [9] ab Seite 99. Es wird wieder von einem bewerteten Graphen $G = (V, E, C(E))$ mit n Knoten ausgegangen, wobei die Kosten c_{ij} für die Kosten der Kante $e = (v_i, v_j)$ stehen. Zusätzlich wird eine Binärvariable $x_{ij} (i, j \in \{1, \dots, n\})$ eingeführt. Diese erhält den Wert 1 sofern die Kante (v_i, v_j) in der Rundreise enthalten ist, sonst den Wert 0. Damit ergibt sich folgende Formulierung:

$$\text{minimiere } \sum_{i=1}^{n} \sum_{j=1}^{n} c_{ij} x_{ij} \qquad [2.1]$$

unter den Nebenbedingungen

$$\sum_{i=1}^{n} x_{ij} = 1 \qquad \text{für alle } j \in \{1, \dots, n\} \qquad [2.2]$$

$$\sum_{j=1}^{n} x_{ij} = 1 \qquad \text{für alle } i \in \{1, \dots, n\} \qquad [2.3]$$

$$\sum_{i \in Q} \sum_{j \in V-Q} x_{ij} \geq 1 \qquad \text{für alle } Q \subset V \; mit \; 2 \leq |Q| \leq \left\lfloor \frac{n}{2} \right\rfloor \qquad [2.4]$$

$$x_{ij} \in \{0,1\} \qquad \text{für alle } i, j \in \{1, \dots, n\} \qquad [2.5]$$

Bei dieser Formulierung gewährleisten die Nebenbedingungen [2.2] und [2.3] dass jeder Knoten genau einmal erreicht und wieder verlassen wird und die Bedingung [2.4], bei der eine Zerlegung der Knotenmenge in Q und $V - Q$ erfolgt, schließt Kurzzyklen, also die Bildung von mehreren unverbundenen Kreisen aus. Dies wird in diesem Fall dadurch erreicht, dass mindestens eine genutzte Kante von einer beliebig gewählten Teilmenge $Q \subset V$ zur Komplementärmenge Q' gefordert wird. Dass auch eine Kante von Q' zurück zu Q gefordert wird ist nicht nötig, da dies durch die beiden Bedingungen [2.2] und [2.3] impliziert wird. Für dieses Ausschließen von Kurzzyklen gibt es verschiedene Formulierungen von Nebenbedingungen wobei die hier gezeigte als Dantzig-Fulkerson-Johnson-Bedingung bekannt ist. Bedingung [2.5] deklariert x_{ij} als binäre Variable.

2.2 Problemdefinition des VRP

Nachdem nun die Problemstellung des TSP formuliert wurde, soll in diesem Abschnitt die Verallgemeinerung des Travelling-Salesman-Problem, das Vehicle Routing Problem in seiner einfachsten Form, dem Capacitated Vehicle Problem (CVRP) veranschaulicht werden, um dadurch der Problemstellung einer realen Tourenplanung einen weiteren Schritt näher zu kommen. Im Zuge dieser Erörterung werden auch die wichtigsten Begriffe, die im weiteren Verlauf dieser Arbeit im Zusammenhang mit der Tourenplanung verwendet werden, definiert. Dabei wird auf die diesbezüglichen Ausführungen in [9] und [44] zurückgegriffen.

Bei dem CVRP handelt sich es um ein Problem das in etwa wie folgt lautet: Von einem Auslieferungslager müssen verschiedene Kunden mit unterschiedlichen (bekannten) Bedarfen beliefert werden. Dafür steht eine beliebig (bzw. ausreichend) große Flotte an homogenen Fahrzeugen, also Fahrzeugen mit denselben Eigenschaften (insbesondere bezüglich der Laderaumkapazität und der entstehenden Kosten), zur Verfügung. Als Ziel wird die Minimierung der Kosten (Gewichtungen) für die benutzten Kanten angenommen, wobei alle Kundenbedarfe als Komplettlieferungen (d.h. eine Aufteilung des Bedarfs eines Kunden auf mehrere Fahrzeuge ist nicht erlaubt) befriedigt werden müssen. Auf die Tatsache, dass sich diese Zielsetzung nicht unbedingt mit der in der Realität meist vorrangigen Zielsetzung der Minimierung der gesamten Transportkosten decken muss, sei an dieser Stelle lediglich mit dem Verweis auf eventuell auftretende Fixkosten je Fahrzeug hingewiesen. Auf die Schwierigkeit, in der Praxis eine geeignete Kostenfunktion zu finden, wird später eingegangen.

Bevor das eben beschriebene Problem genauer erörtert wird, sollen zunächst die wichtigsten Begriffe im Zusammenhang mit der Tourenplanung bzw. dem Vehicle Routing Problem definiert werden: Als **Depot** wird jener Ort bezeichnet, an dem die einzelnen Fahrten beginnen bzw. enden, also im obigen Beispiel das Auslieferungslager. Die Bedarfe der einzelnen Kunden werden als **Aufträge** bezeichnet und die Menge an Aufträgen, die mit demselben Fahrzeug bedient werden wird als **Tour** definiert. Die Reihenfolge in der das Fahrzeug die einzelnen Aufträge erledigt, wird **Route** der Tour genannt. Eine Lösung des gesamten Problems, also die Zusammenstellung der Touren mit den dazugehörigen Routen, die alle Bedingungen des Problems erfüllt, wird als **Tourenplan** bezeichnet.

Anhand dieser kurzen Definition der wichtigsten Begriffe kann bereits eine wesentliche Charakteristik aller Vehicle Routing Probleme erkannt werden, nämlich das Zusammenspiel zweier grundlegender Problemstellungen, dem **Clustering** und dem **Routing**. Das Clustering entspricht der Aufteilung der Aufträge auf die einzelnen Touren und durch das Routing wer-

den die einzelnen Routen erstellt. Damit ist auch der direkte Zusammenhang des Vehicle Routing Problems mit dem Travelling-Salesman-Problem gezeigt, denn dieses Routing einer Tour entspricht genau der Problemstellung des TSP, wodurch das TSP ein Teilproblem des VRP wird und umgekehrt kann das TSP als VRP gesehen werden, bei dem ein einziges Fahrzeug zur Befriedigung der gesamten Kundenbedarfe ausreicht. Der Zusammenhang zwischen dem Clustering und dem Routing innerhalb der Tourenplanung ist nochmals in Abbildung 1 schematisch dargestellt (Vgl. Abbildung in [44], S.41).

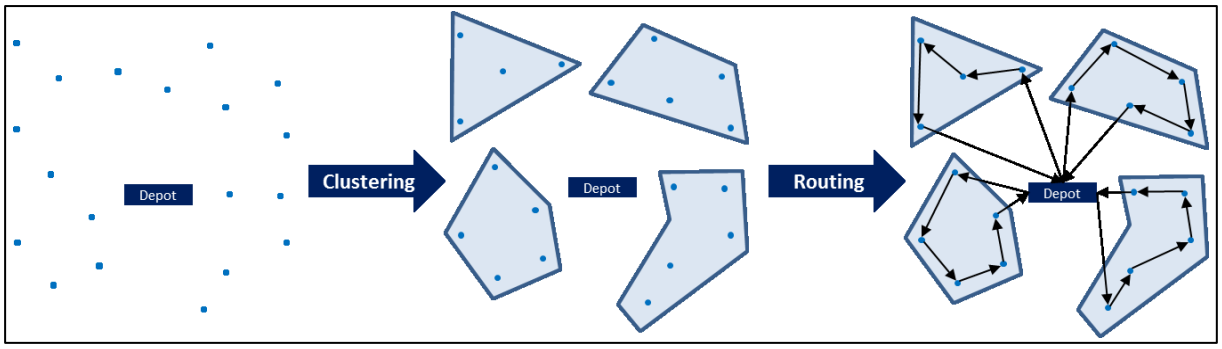


Abbildung 1: Tourenplanung - Clustering/Routing

Wie schon das TSP soll nun an dieser Stelle das CVRP als ganzzahliges lineares Optimierungsproblem beschrieben werden. Die Literatur bietet hierfür eine Vielzahl an Möglichkeiten die sich meist in der Berücksichtigung der Nebenbedingungen unterscheiden. Einen Überblick über diese bieten Toth und Vigo in „An Overview of Vehicle Routing Problems" [41]. An dieser Stelle wird eine Formulierung gewählt, auf die im weiteren Verlauf dieser Arbeit aufgebaut werden kann und die sich leicht an Erweiterungen des VRP anpassen lässt. Dabei wird auf die Formulierungen in [3], [41] und [44] zurückgegriffen.

Die oben beschriebene Problemstellung des CVRP lässt sich folgendermaßen beschreiben: Wie bei dem TSP ist ein bewerteter Graph $G = (V, E, C(E), B(V))$ gegeben, der jedoch um die Menge $B(V)$, die die Bewertungen der Knoten, d.h. die Bedarfe der einzelnen Kunden enthält, ergänzt wird. Die Anzahl der Kunden wird mit n angegeben und die Knoten werden mit $v_0, v_1, \dots, v_n$ bezeichnet, wobei der Knoten v_0 dem Depot entspricht. Der Bedarf eines Kunden $b(v_i)$ wird verkürzt als b_i bezeichnet und die Gewichtung einer Kante $c(e_{ij})$ wird, wie bereits bei der Formulierung des TSP, verkürzt als c_{ij} angegeben. Für die Bedienung der Kunden steht eine ausreichend hohe Anzahl an gleichen Fahrzeugen zur Verfügung die mit $k = 1, \dots K$ nummeriert werden, wobei K der Anzahl der verwendeten Fahrzeuge entspricht. Jedem Fahrzeug ist eine maximale Kapazität Q_k zugeordnet, wobei im Falle des einfachen CVRP gilt, dass alle $Q_k, k \in \{1, \dots K\}$, gleich sind. Zusätzlich zu der Binärvariable x_{ijk}, die ähnlich wie bei der Beschreibung des TSP nur dann den Wert 1 annimmt, wenn die Kante e_{ij} vom Fahrzeug k genutzt wird, wird die Binärvariable y_{ik} eingeführt, die dann auf 1 gesetzt wird, wenn das

Fahrzeug k den Knoten i bedient. In allen übrigen Fällen nehmen die Binärvariablen den Wert 0 an. Die Menge an Knoten, die von Fahrzeug k angefahren wird, wird als V_k bezeichnet. Unter Berücksichtigung des anfangs erwähnten Ziels der Minimierung der Gesamtkosten, die in diesem Modell als Summe der Kosten aller genutzten Kanten dargestellt wird, ergibt sich folgende Formulierung:

$$\text{minimiere } \sum_{i=0}^{n} \sum_{j=0}^{n} \sum_{k=1}^{K} c_{ij} x_{ijk} \qquad [2.6]$$

unter den Nebenbedingungen

$$y_{0k} = 1 \qquad \text{für alle } k \in \{1, \dots, K\} \qquad [2.7]$$

$$\sum_{k=1}^{K} y_{ik} = 1 \qquad \text{für alle } i \in \{1, \dots, n\} \qquad [2.8]$$

$$\sum_{i=1}^{n} y_{ik} b_i \leq Q_k \qquad \text{für alle } k \in \{1, \dots, K\} \qquad [2.9]$$

$$y_{ik} \in \{0,1\} \qquad \text{für alle } k \in \{1, \dots, K\}; i \in \{1, \dots, n\} \qquad [2.10]$$

sowie für alle $k \in \{1, \dots, K\}$:

$$\sum_{i \in V_k}^{n} x_{ijk} = 1 \qquad \text{für alle } j \in V_k \qquad [2.11]$$

$$\sum_{j \in V_k}^{n} x_{ijk} = 1 \qquad \text{für alle } i \in V_k \qquad [2.12]$$

$$\sum_{i \in Q} \sum_{j \in V_k - Q} x_{ijk} \geq 1 \qquad \text{für alle } Q \subset V_k \; mit \; 2 \leq |Q| \leq \left\lfloor \frac{\sum_{i=0}^{n} y_{ik}}{2} \right\rfloor \qquad [2.13]$$

$$x_{ijk} \in \{0,1\} \qquad \text{für alle } i, j \in \{1, \dots, n\} \qquad [2.14]$$

In dieser Formulierung gewährleisten die Nebenbedingungen [2.7] bis [2.10] ein zulässiges Clustering. Dabei stellt [2.7] sicher, dass jede Tour das Depot beinhaltet, [2.8] fordert die bereits erwähnten Komplettlieferungen und [2.9] berücksichtigt die maximale Kapazität der einzelnen Fahrzeuge. In [2.10] wird y_{ik} als Binärvariable deklariert. Die Nebenbedingungen [2.11] bis [2.14] stellen die vom TSP bereits bekannten Forderungen für ein zulässiges Routing jeder einzelnen Tour dar.

Wie bereits mehrfach erwähnt, ist die hier gezeigte Problemstellung lediglich das Grundmodell des Vehicle Routing Problems und es gibt mittlerweile eine Vielzahl von daraus abgeleiteten Standardproblemen, auf die im Abschnitt 2.4 eingegangen wird. Davor soll jedoch ein Überblick über die wichtigsten Merkmale, in denen sich Problemstellungen in der Praxis unterscheiden können, gegeben werden.

2.3 Unterscheidungsmerkmale von VRPs

Die Problemstellungen innerhalb der Tourenplanung können sich in der Praxis teils sehr stark unterscheiden, da meist branchen- bzw. unternehmensspezifische Kriterien zu berücksichtigen sind. Diese Unterschiede lassen sich in die vier in Abbildung 2 gezeigten Kategorien, die sich aus der Art der Auswirkung auf die oben beschriebene Basis-Problemstellung ergeben, unterteilen: Modifikation der Aufgabenstellung, Ausweitung der Möglichkeiten, Erweiterung der Restriktionen und Veränderung der Zielsetzung. Davon abweichende Einteilungsmöglichkeiten sowie zusätzliche Beispiele zu den unten angeführten, können [9] ab Seite 200 und [44] ab Seite 46 entnommen werden.

	Modifikation der Aufgabenstellung	Ausweitung der Möglichkeiten	Erweiterung der Restriktionen	Veränderung der Zielsetzung
Auswirkung:	**Veränderung der gesamten Problemstruktur**	**Vergrößerung des Lösungsraums**	**Reduktion der Anzahl an zulässigen Lösungen**	**Anpassung der Zielfunktion**
Beispiele:	- Bedarfe fallen an den Kanten an - Pickup and Delivery-Probleme - ...	- Erhöhung der Anzahl an Depots - Teilbarkeit von Aufträgen - Heterogene Flotte an Fahrzeugen - ...	- Zeitfenster - Angabe maximaler Tourdauer - Kunden-Fahrzeug-Relationen - ...	- Erweiterung der Kostenfunktion - Mehrdimensionale Zielsetzung (z.B. Kosten und Termintreue) - ...

Abbildung 2: Kategorien der Abänderung vom CVRP

Eine **Modifikation der Aufgabenstellung** stellt natürlich die größte Abweichung vom Basisproblem dar und geht meist mit einer starken Veränderung der gesamten Formulierung der Problemstellung einher. Solch eine Modifikation wäre beispielsweise, wenn der Bedarf nicht an den Knoten sondern an den Kanten anfällt, wie es unter anderem im Bereich der Abfallentsorgung der Fall ist. Ein weiteres Beispiel für diese Art der Veränderung ist das sogenannte Pickup and Delivery Problem bei dem an den Knoten sowohl Abholungen als auch Belieferungen durchzuführen sind (z.B. Auslieferung von Getränken und gleichzeitiges Einsammeln von Leergut). Da sich das in dieser Arbeit konkret behandelte Problem auf ein reines Auslieferungsproblem reduzieren lässt und es auch keine andersartige Modifikation der Aufgabenstellung beinhaltet, wird im weiteren Verlauf dieser Arbeit auf solche Ausprägungen des VRP nicht weiter eingegangen.

Durch eine **Ausweitung der Möglichkeiten** wird die Anzahl an verschiedenen zulässigen Tourenplänen erhöht und wirkt sich somit vergrößernd auf den potenziellen Lösungsraum aus. Diese Ausweitungen lassen sich meist entweder der Depotstruktur, der Auftrags- bzw. Kundenstruktur oder der Fahrzeugstruktur zuordnen. Das wichtigste Beispiel für solch eine Veränderung der Depotstruktur wäre die Erhöhung der Anzahl an Depots, wie es der Fall ist,

wenn mehrere Auslieferungslager mit demselben Sortiment für die Befriedigung der Kundenbedarfe zur Verfügung stehen. Eine lösungsraumvergrößernde Veränderung der Auftragsstruktur wäre es, wenn zum Beispiel der Bedarf eines Kunden auf mehrere Fahrzeuge aufgeteilt werden kann. Bezüglich der Fahrzeugstruktur wäre die Möglichkeit einer heterogenen Flotte an Fahrzeugen, d.h. Fahrzeugen mit unterschiedlichen Charakteristiken (insbesondere der Kapazität) zu nennen.

Im Gegensatz zur Ausweitung der Möglichkeiten verringert eine **Erweiterung der Restriktionen** natürlich die Anzahl an zulässigen Tourenplänen. Als wichtigste Vertreter dieser Gattung sind sicherlich die zeitbasierte Einschränkungen, meist in Form von sogenannten Zeitfenstern zu nennen. Solche Einschränkungen, bei denen Kunden nur zu bestimmten Zeiten bedient werden können (oder sollten) kommen in der Praxis sehr häufig vor, weshalb auf diese im nächsten Abschnitt im Speziellen eingegangen wird. Eine andere, in der Praxis häufig vorkommende zeitbasierende Einschränkung, ist die Vorgabe einer maximalen Dauer für die einzelnen Touren. Als Beispiel für eine nicht zeitbasierte Einschränkung wäre der Fall zu nennen, dass nicht jeder Kunde von jedem zur Verfügung stehenden Fahrzeug bedient werden kann. Dies ist zum Beispiel der Fall, wenn manche Kunden nur mit kleineren Transportmitteln erreichbar sind (etwa im Zentrum von Städten) oder zur Entladung bestimmte Vorrichtungen notwendig sind.

Die letzte wesentliche Möglichkeit, wie sich verschiedene VRPs vom im vorigen Abschnitt erörterten Basis-Problem unterscheiden können, ist eine **Veränderung der Zielsetzung**. Es wurde bereits im vorigen Abschnitt kurz angedeutet, dass die simple Minimierung der Summe aller Gewichtungen der benutzten Kanten, das heißt in der Regel die Summe der Entfernungen, nicht immer den Ansprüchen in der Praxis genügen. Ein wichtiges Beispiel dafür ist es, wenn die Kostenfunktion, die es zu minimieren gilt, noch weitere Komponenten, wie zum Beispiel die bereits kurz erwähnten Fixkosten je genutztem Fahrzeug, enthält. Eine noch weit schwieriger zu berücksichtigende Veränderung der Zielsetzung ist es, wenn neben den Transportkosten auch noch andere Ziele, wie zum Beispiel die Wunschtermintreue oder Ähnliches verfolgt werden müssen. Da für die Entwicklung einer guten Lösung eines speziellen VRPs die Formulierung einer geeigneten Kostenfunktion und die ausgewogene Berücksichtigung aller Ausprägungen der Zielsetzung von hoher Bedeutung sind, wird auf den Umgang mit den innerhalb der Tourenplanung häufig anzutreffenden Zielsetzungen im Abschnitt 2.5 näher eingegangen.

Anhand der hier gezeigten Beispiele kann erkannt werden, dass die Problemstellungen von in der Praxis auftretenden VRPs sehr vielseitig sein können, wodurch sich bereits mehrere

Standarderweiterungen des CVRP ergeben haben. Auf diese wird im folgenden Abschnitt eingegangen.

2.4 Das VRPTW und weitere wichtige Erweiterungen des Basisproblems

An dieser Stelle werden nun die wichtigsten Standarderweiterungen des Basisproblems kurz erörtert, wobei auf das Vehicle Routing Problem with Time Windows (VRPTW),auch bekannt als Vehicle Routing and Scheduling Problem (VRSP), im Besonderen eingegangen wird. Diese Erweiterung wird deshalb so hervorgehoben, da diese erstens die in der Praxis wahrscheinlich am häufigsten vorkommende Erweiterung darstellt (und auch in dem in dieser Arbeit untersuchten Problem eine Rolle spielt) und zweitens diese Erweiterung meist umfangreiche Anpassungen von bekannten Algorithmen und Heuristiken bzw. Metaheuristiken für die Suche nach Lösungen erfordert (wie im nächsten Kapitel gezeigt wird).

Bei dem VRPTW wird die Basisproblemstellung aus Abschnitt 2.2 dahingehend erweitert, dass jedem Kunden bestimmte Zeitfenster zugeordnet sind, in denen dieser bedient werden kann. Dabei wird grundsätzlich zwischen harten Zeitfenstern, die unbedingt eingehalten werden müssen und weichen Zeitfenstern, die nicht zwingend eingehalten werden müssen, jedoch bei Nichtbeachtung Strafkosten verursachen, unterschieden. Da es sich in dem in dieser Arbeit untersuchtem Beispiel um harte Zeitfenster handelt und weiche Zeitfenster eigentlich lediglich eine Veränderung der Zielsetzung, also die Erweiterung der Zielfunktion um die Dimension der Termintreue, darstellen, werden im weiteren Verlauf dieser Arbeit nur harte Zeitfenster als Zeitfenster bezeichnet und somit auf die Angabe, dass es sich dabei um zwingend einzuhaltende handelt, verzichtet.

Um solche Zeitfenster zu berücksichtigen, muss die Formulierung des Problems von Abschnitt 2.2 angepasst bzw. erweitert werden, wobei auf den Ausführungen in ([3], S.254ff) aufgebaut wird. Obwohl in der Literatur zu diesem Thema meist nur Formulierungen des Problems zu finden sind, bei denen jedem Kunden nur ein Zeitfenster zugeordnet ist (und auf die Möglichkeit, dass es mehrere geben kann, nur hingewiesen wird), soll an dieser Stelle gleich eine Möglichkeit gezeigt werden, die auch mehrere Zeitfenster je Kunden zulässt. Zunächst wird die Problembeschreibung des CVRP wie folgt ergänzt:

Jedem Kunden i ist eine (nicht leere) Menge an Zeitfenstern $M_i = \{(e_{i,1}, l_{i,1}), \dots, (e_{i,A_i}, l_{i,A_i})\}$ zugeordnet wobei mit e_{i,z_i} und l_{i,z_i} die untere bzw. obere Schranke des Zeitfensters z_i des

Kunden i angegeben wird. A_i gibt die Gesamtanzahl an Abladezeitfenstern des Kunden i an. Weiters wird jedem Kunden neben der Bewertung des Bedarfs b_i auch eine Servicezeit s_i zugeteilt, die angibt, wie lange die Bedienung des jeweiligen Kunden dauert. Diese Bedienung muss zur Gänze innerhalb eines der angegebenen Zeitfenster erfolgen, weshalb nur Zeitfenster zulässig sind, in denen eine gesamte Servicierung möglich ist. Liegt die Ankunftszeit a_{ik} eines Fahrzeugs k außerhalb dieser Zeitfenster des jeweiligen Kunden, oder reicht die verbleibende Zeit des aktuellen Zeitfensters für die gesamte Bedienung nicht aus, so muss auf den Beginn des nächsten Zeitfensters gewartet werden und es entsteht bei diesem Kunden i eine Wartezeit w_i. Die Abfahrtszeit a_{0k} vom Depot kann für jede Tour k frei gewählt werden. Als Ziel wird bei der einfachsten Form des VRPTW die Minimierung der Gesamtdauer aller Touren definiert. Dafür wird jeder Kante (v_i, v_j) eine Fahrzeit t_{ij} als Bewertung zugeordnet. Damit ergibt sich folgende Formulierung der Problemstellung:

$$\text{minimiere } \sum_{i=0}^{n}\sum_{j=0}^{n}\sum_{k=1}^{K} c_{ij}x_{ijk} \qquad \text{mit } c_{ij} = t_{ij} + w_i + s_i \qquad [2.15]$$

unter den Nebenbedingungen [2.7] bis [2.14] (Abschnitt 2.2) sowie

$$l_{i,z_i} - e_{i,z_i} \geq s_i \qquad \text{für alle } i \in \{1,\dots,n\} \text{ und } z_i \in \{1,\dots,A_i\} \qquad [2.16]$$

$$l_{i,z_i} < l_{i,z_i+1} \qquad \text{für alle } i \in \{1,\dots,n\} \text{ und } z_i \in \{1,\dots,A_i - 1\} \qquad [2.17]$$

$$a_{ik} + s_i \leq l_{i,A_i} \qquad \text{für alle } k \in \{1,\dots,K\};\ i \in \{1,\dots,n\} \qquad [2.18]$$

$$a_{jk} = \sum_{k=1}^{K}\sum_{i=0}^{n} x_{ijk}(a_{ik} + w_i + s_i + t_{ij}) \qquad \text{für alle } j \in \{1,\dots,n\} \qquad [2.19]$$

$$a_{ik}, w_i, s_i \geq 0 \qquad \text{für alle } k \in \{1,\dots,K\};\ i \in \{0,\dots,n\} \qquad [2.20]$$

Die Nebenbedingungen [2.16] bis [2.20] gewährleisten die Einhaltung der zeitlichen Restriktionen wobei [2.16] eine für die Servicierung des jeweiligen Kunden ausreichende Größe jedes Zeitfensters verlangt und [2.17] zeigt die Chronologie der Zeitfenster. Durch [2.18] wird gefordert, dass jeder Kunde vor dem Schließen seines letzten Zeitfensters erreicht wird und dass auch die gesamte Servicierung innerhalb dieses Zeitfensters erfolgen kann (da durch [2.16] eine ausreichende Größe des Zeitfensters vorausgesetzt werden kann). [2.19] stellt die zeitliche Abfolge der einzelnen Routen dar, indem sich die Ankunftszeit jedes Kunden durch die Ankunftszeit des auf der Route davorliegenden Knoten plus dessen Warte- und Servicezeit und der Fahrzeit von diesem Knoten zu dem Kunden ergibt. Dass für die Berechnung der Ankunftszeit jedes Kunden tatsächlich nur die relevanten Zeiten des in der Route direkt davorliegenden Knoten berücksichtigt werden, wird dadurch gewährleistet, dass die Binärvariable x_{ijk} für ein bestimmtes j nur dann den Wert 1 annimmt, wenn innerhalb der Tour k der Knoten j direkt nach dem Knoten i angefahren wird. Dies kann auf Grund der

Nebenbedingungen [2.7] sowie [2.11] und [2.12] nur genau einmal der Fall sein. Die letzte Nebenbedingung [2.20] schließt negative Ankunfts-, Warte- und Servicezeiten aus.

Wie bereits erwähnt, gibt es neben dem VRPTW noch zahlreiche andere Erweiterungen des VRP die sich durch Veränderungen, wie sie im vorigen Abschnitt beschrieben wurden, ergeben. Als wichtige Beispiele seien das Multi Depot Vehicle Routing Problem (MDVRP), bei dem mehrere Depots zur Verfügung stehen, das Heterogeneous Fleet Vehicle Routing Problem (HFVRP), bei dem es verschiedene Fahrzeugtypen gibt, und das Open Vehicle Routing Problem (OVRP), bei dem die Fahrzeuge nicht wieder zum Depot zurückkehren müssen, genannt. An diesen Beispielen ist zu sehen, dass sich die Konvention, alle Erweiterungen des Standardproblems ebenfalls als Akronym zu bezeichnen, etabliert hat. Da es natürlich auch Problemstellungen gibt, bei denen eine Kombination aus mehreren Erweiterungen auftritt (und dabei sichtlich versucht wird, diese Konvention beizubehalten), treten in der Literatur auch etwas längere Akronyme wie HFVRPTWNPD für Heterogeneous Fleet Vehicle Routing Problem with Time Windows and Nonlinearly Penalized Delays auf [39].

2.5 Zielsetzungen von VRPs innerhalb der Tourenplanung

Bis jetzt wurde als Ziel der einzelnen Problemstellungen entweder die Minimierung der Summe über die Bewertungen aller genutzten Kanten (also in der Regel die Minimierung der gesamt zurückgelegten Distanz) oder die Minimierung der insgesamt benötigten Zeit angegeben. Bevor nun im nächsten Kapitel auf konkrete Lösungsmöglichkeiten für VRPs eingegangen wird, soll an dieser Stelle erörtert werden, welche grundlegenden Ziele innerhalb der Tourenplanung verfolgt werden und wie diese innerhalb einer Lösungsfindung berücksichtigt werden können.

Als die vier wesentlichen Ziele innerhalb der Tourenplanung, die in jedem Unternehmen unterschiedlich gewichtet werden, sind die Maximierung des Lieferservices, die Minimierung der Transportkosten, die Minimierung von Umweltbelastungen sowie die Maximierung der Mitarbeiterzufriedenheit zu nennen. Diese leiten sich natürlich von den übergeordneten Unternehmenszielen, wie zum Beispiel Sicherung des Fortbestands des Unternehmens und Gewinnmaximierung ab [44].

Da bei der Lösungssuche der Vergleich zwischen zwei Tourenplänen hinsichtlich ihres Zielerreichungsgrads essentiell ist, müssen für die oben angeführten Fundamentalziele quantifizier- und somit vergleichbare Eigenschaften von Tourenplänen definiert werden, die einen

Rückschluss auf das Ausmaß der Zielerreichung zulassen. Die dadurch abgeänderten Ziele, wie das oben erwähnte Minimieren der gesamt zurückgelegten Strecke, werden als Instrumentalziele bezeichnet. Abbildung 3 zeigt die vier wichtigsten und am häufigsten verwendeten Instrumentalziele von VRPs und deren Einfluss auf die vorher genannten Ziele innerhalb der Tourenplanung (Vgl. Abbildung in [44], Seite 88). Wie bereits mehrfach angedeutet, gibt es in der Praxis auch häufig Fälle, in denen mehrere Ziele gleichzeitig verfolgt werden, weshalb bei der Lösungssuche auch öfters mehrere dieser Instrumentalziele Verwendung finden. Wie solch eine Lösungssuche mit einer multikriteriellen Zielsetzung erfolgen kann, wird im nächsten Kapitel beschrieben.

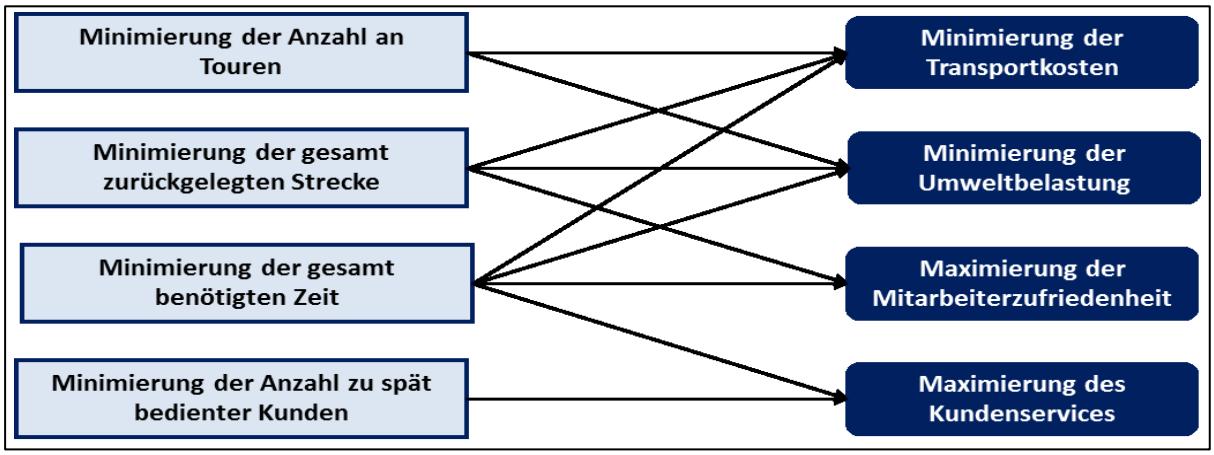


Abbildung 3: Wichtige Instrumentalziele innerhalb der Tourenplanung

3 Bekannte Ansätze für das Lösen von VRPs

Nachdem nun das Vehicle Routing Problem bzw. die Problemstellungen innerhalb der Tourenplanung beschrieben wurden, soll in diesem Kapitel ein Überblick über die bekanntesten (und vielversprechendsten) Lösungsansätze für VRPs und deren Erweiterungen gegeben werden. Dies wird auch als Basis für die Erarbeitung einer Lösung des später konkret behandelten Problems dienen. Da, wie bereits kurz erwähnt, lediglich sehr kleine Probleme dieser Art, d.h. zum Beispiel Probleme mit einer geringen Anzahl an Kunden, exakt gelöst werden können, wird auf solche Verfahren in dieser Arbeit nicht näher eingegangen. Interessierte Leser finden hierfür Beschreibungen diverser Branch-and-Bound bzw. Branch-and-Cut Algorithmen unter anderem in [42].

Im ersten Abschnitt dieses Kapitels wird eine Möglichkeit zur Einteilung der verschiedenen Lösungsansätze bzw. Heuristiken im Allgemeinen, sowie im Speziellen für Tourenplanungsprobleme, beschrieben. Anschließend wird auf die Lösung von TSPs eingegangen, und danach werden die wichtigsten VRP-spezifischen Heuristiken gezeigt. Darauf folgt eine Beschreibung der wichtigsten Metastrategien die auf das Lösen von VRPs angewendet werden. Am Ende des Kapitels sollen zu den hier gezeigten Lösungsansätzen publizierte Ergebnisse zusammengefasst und verglichen werden, um daraus bereits vielversprechende Vorgehensweisen für die Lösung des vorliegenden Problems ableiten zu können.

3.1 Klassifikation von Heuristiken

Auf Grund der beschränkten Einsatzfähigkeit von deterministischen Verfahren auf reale Tourenplanungsprobleme, wird der Entwicklung von heuristischen Verfahren für VRPs eine große Aufmerksamkeit geschenkt. Solche Heuristiken stellen im Gegensatz zu deterministischen Verfahren nicht den Anspruch garantiert eine optimale Lösung zu finden, bzw. eine gefundene optimale Lösung als solche zu erkennen. Ihr Ziel ist es, auch bei sehr großen Problemen noch in angemessener Zeit eine gute (vielleicht sogar optimale) Lösung zu finden - diese Eigenschaft macht solche Verfahren in der Praxis so bedeutend. Im Gegensatz zu Metaheuristiken (bzw. Metastrategien), die durch geringe Anpassungen auf verschiedenste Problemstellungen angewendet werden können, sind Heuristiken in der Regel auf ein bestimmtes Problem zugeschnitten. Des Öfteren sind jedoch solche Heuristiken direkt von Metastrategien abgeleitet.

Im Allgemeinen lassen sich heuristische Verfahren auf Grund ihrer generellen Vorgangsweise bzw. ihres Zwecks in die fünf Gruppen Eröffnungsverfahren, lokale Such- bzw. Verbesserungsverfahren, populationsbasierte Verfahren, relaxationsbasierte Verfahren sowie unvollständig ausgeführte exakte Verfahren unterteilen [9]:

Eröffnungsverfahren dienen der Konstruktion einer ersten zulässigen Lösung und werden in der Regel als Ausgangspunkt für weitere Verbesserungen herangezogen. Beispiele für solche Verfahren sind die Nearest-Neighbour-Heuristik und das sukzessive-Konstruktions-Verfahren, die im nächsten Abschnitt beschrieben werden. Bei solchen Verfahren muss im Allgemeinen mit steigendem Anspruch an die Lösungsgüte, ein höherer Rechenaufwand in Kauf genommen werden.

Lokale Such- bzw. Verbesserungsverfahren dienen der sukzessiven Verbesserung einer Startlösung (die meist durch ein Eröffnungsverfahren generiert wird), und folgen einem iterativen Schema. Dabei werden durch geringfügige Änderungen der Anfangslösung sogenannte Nachbarschaftslösungen generiert, und deren (unter Umständen nur geschätzten) Auswirkungen auf den Zielfunktionswert verglichen. Bringt eine Nachbarschaftslösung eine Verbesserung mit sich, so wird diese als neue Ausgangslösung herangezogen und der Vorgang wiederholt sich. Bei der Auswahl der verbesserten Nachbarschaftslösung kann zwischen der „First-Fit"-Methode, bei der die erste Lösung, die eine Verbesserung mit sich bringt, gewählt wird, und der „Best-Fit"-Methode, bei der aus allen Nachbarschaftslösungen diejenige mit der größten Verbesserung gewählt wird, unterschieden werden. Soll durch solch ein Verfahren nicht nur ein lokales Optimum gefunden werden, so müssen auch zeitweise Verschlechterungen akzeptiert werden. Wann welche Verschlechterungen akzeptiert werden wird meist durch eine Metastrategie, wie Tabu Search (siehe Abschnitt 3.4.2) oder Simulated Annealing, entschieden. Beispiele für ein reines Verbesserungsverfahren sind das k-opt-Verfahren (siehe Abschnitt 3.2.2) oder der Savings-Algorithmus (Abschnitt 3.3.1).

Populationsbasierende Verfahren benötigten im Unterschied zur lokalen Suche nicht nur eine, sondern mehrere Ausgangslösungen. Diese werden dann in unterschiedlicher Form kombiniert, wodurch neue Lösungen geschaffen werden, die wiederum eine neue Ausgangspopulation für weitere Kombinationen darstellen. Dabei unterscheiden sich die Verfahren insbesondere in der Vorgangsweise der Selektion der Population sowie der Kombination der verschiedenen Lösungen. Die wichtigsten Vertreter solcher Verfahren sind genetische Algorithmen und Ameisenalgorithmen (siehe Abschnitte 3.4.3 und 3.4.4) – beide auf Grund ihrer vielfältigen Einsatzmöglichkeiten den Metaheuristiken zuzuordnen.

Bei **relaxationsbasierten Verfahren** wird zunächst das Ausgangsproblem vereinfacht, und für das abgeleitete Problem (der Relaxation des Ausgangsproblems) eine optimale, oder zumindest gute, Lösung gesucht. Anschließend wird die so gefundene Lösung so weit abgeändert, dass sie auch eine zulässige Lösung des Ausgangsproblems darstellt. Eine Möglichkeit der Relaxation ist die LP-Relaxation, bei der aus einem ganzzahligen Optimierungsproblem ein leichter lösbares reelles Optimierungsproblem gemacht wird. Dessen Lösung muss anschließend durch Runden (oder anderen Anpassungen) wieder zu einer zulässigen Lösung für das Ausgangsproblem gemacht werden. Eine andere Art der Relaxation ist das Weglassen von Nebenbedingungen, wobei es dabei auch die Möglichkeit gibt, diese in entsprechender Form in die Zielfunktion aufzunehmen. Solch eine Relaxation wird als Lagrange-Relaxation bezeichnet.

Zu der letzten Gruppe der Heuristiken, den **unvollständig ausgeführten exakten Verfahren**, zählen alle exakte Verfahren, die vorzeitig abgebrochen werden. Da in dieser Arbeit aber nicht auf exakte Verfahren eingegangen wird, werden auch diese Heuristiken nicht näher erörtert.

Wie im vorigen Kapitel gezeigt wurde, gibt es bei dem VRP zwei Teilprobleme zu lösen: das Clustering und das Routing. Auf Grund dieser Eigenschaft ergibt sich für Heuristiken, die auf das VRP angewandt werden, eine weitere Einteilungsmöglichkeit: Werden die zwei Teilprobleme hintereinander gelöst, so ist die Heuristik den **Sukzessivverfahren** zuzuordnen, werden jedoch das Clustering und das Routing gleichzeitig durchgeführt, so handelt es sich um ein **Simultanverfahren**. Die Sukzessivverfahren können des Weiteren dahingehend klassifiziert werden, in welcher Reihenfolge die zwei Teilprobleme behandelt werden. Wird zunächst eine gesamte Rundreise erstellt, also ein TSP als Relaxation des VRP gelöst, und erst anschließend diese Gesamtroute auf Touren aufgeteilt, so ist dies ein „**route first – cluster second – Verfahren**". Im umgekehrten Fall, dem „**cluster first – route second – Verfahren**", werden die Aufträge zunächst auf Touren aufgeteilt, und danach für die einzelnen Touren die Bedienreihenfolge festgelegt ([9], S.226f).

3.2 Heuristiken zur Lösung von TSPs

Wie bereits bei der Problembeschreibung, soll auch bei der Beschreibung der möglichen Lösungen mit dem Travelling-Salesman Problem begonnen werden. Abgesehen von „route first – cluster second – Verfahren" tritt dieses bei VRPs lediglich als Routing der einzelnen Touren

auf. Diese Touren besitzen auf Grund der Kapazitätsbeschränkung des Fahrzeugs selten eine große Anzahl an Abladestellen – im später gezeigten Beispiel geht diese nur in Ausnahmefällen über 30. Deshalb werden bei VRPs in der Regel keine aufwendigen, rechenzeitintensiven Heuristiken für das Routen benötigt. Hier ist es vielmehr von Bedeutung, bei Veränderungen der Zuteilung von Kunden zu Touren, möglichst schnell eine qualifizierte Auskunft über die zu erwartende Veränderung der Distanz oder der Dauer geben zu können. Eine entsprechende (rechenzeitintensivere) Optimierung der einzelnen Routen erfolgt üblicherweise nur für den am Ende ausgewählten Tourenplan. Aus diesem Grund werden in diesem Abschnitt lediglich die zwei wichtigsten Eröffnungs- bzw. Konstruktionsheuristiken sowie eine Möglichkeit zur lokalen Suche gezeigt. Bei größeren TSPs (bzw. für „Route first – cluster second – Verfahren") kann auch auf dieselben Metaheuristiken zurückgegriffen werden, die auch bei VRPs Anwendung finden (Abschnitt 3.4).

Im weiteren Verlauf wird bei den Beschreibungen der einzelnen Verfahren häufig von Rundreisen gesprochen, dies wird als Synonym für Kreise genommen. Auf die Tatsache, dass sich diese Verfahren aber auch auf offene Routen, also Wege, anwenden lassen, wird hier einmalig hingewiesen (und in den Abbildungen gezeigt).

3.2.1 Eröffnungsheuristiken

Die zwei wichtigsten, und sicherlich meist genutzten Eröffnungs- bzw. Konstruktionsheuristiken für TSPs sind die Nearest-Neighbor-Heuristik und die sukzessive Konstruktion.

Bei der **Nearest-Neighbor-Heuristik** wird bei dem Bilden einer Rundreise mit einem beliebigen Knoten begonnen, und anschließend jeweils der nächstgelegene Knoten zur Route hinzugefügt. Wurden alle Knoten in die Route aufgenommen, wird die Rundreise geschlossen. Soll keine Rundreise, sondern eine offene Route mittels diesem Verfahren erstellt werden, so muss natürlich bei dem vorgegebenen Startpunkt (z.B. bei dem Depot) begonnen werden. Der Vorteil dieses Verfahrens ist, dass es sehr einfach (und damit auch schnell) passable Lösungen liefert, die sich gut als Anfangslösungen für lokale Suchverfahren eignen. Ein wesentlicher Nachteil ist sicherlich, dass Restriktionen, wie zum Beispiel Zeitfenster, nicht berücksichtigt werden können.

Bereits etwas bessere Ergebnisse als mit der Nearest-Neighbor-Heuristik können mittels einer **sukzessiven Konstruktion** erzielt werden. Bei dieser Heuristik wird ausgehend von einer Startroute mit zwei Knoten, für jeden weiteren Knoten die jeweils beste Einfügestelle in der bereits konstruierten Route gewählt. Dabei ist die gefundene Lösung natürlich von den zwei Startknoten, sowie der Reihenfolge, in der die Knoten in die Tour einbezogen werden,

abhängig. Hierfür wird unter anderem in [9] eine Vorgangsweise, bei der mit zwei weit entfernten Knoten begonnen wird, empfohlen. Bei dieser Vorgangsweise soll anschließend jeweils der Knoten als nächstes einbezogen werden, der von der bisherigen Route am weitesten entfernt ist. Der Vorteil gegenüber der Nearest-Neighbor Heuristik, in der Regel bessere Lösungen zu finden, wird natürlich mit längeren Rechenzeiten erkauft.

3.2.2 lokale Suche mittels λ-opt-Verfahren

Aufbauend auf einer durch ein Eröffnungsverfahren generierten Lösung, wird diese meist durch ein lokales Suchverfahren noch verbessert. Für TSPs kommt hierfür sicherlich am häufigsten ein λ-opt-Vefahren (oder eine Abwandlung davon) zum Einsatz, weshalb dieses näher erörtert wird. Bei solch einem reinen Verbesserungsverfahren, bei dem λ für die Anzahl zu vertauschender Kanten steht, werden so lange λ Kanten gegen λ andere Kanten getauscht, bis dadurch keine Verbesserung mehr erzielt werden kann. Eine Route, auf die dieses Verfahren angewandt wurde, wird λ-optimal genannt, d.h. diese Rundreise oder Strecke kann durch den Austausch von λ Kanten nicht weiter verbessert werden. Weiters gilt, dass diese Route k-optimal für alle k≤λ ist, und eine n-optimale Route für n Knoten immer die optimale Reihenfolge definiert ([9], S.111).

In Abbildung 4 ist ein Schritt dieser Heuristik im einfachsten Fall, dem 2-opt-Verfahren, gezeigt. Dabei werden zwei Kanten aus der Route entfernt und durch zwei neue ersetzt, wobei wieder eine zusammenhängende Route entstehen muss. Hierfür gibt es beim 2-opt-Vefahren lediglich eine zulässige Möglichkeit. Zu beachten ist, dass sich bei solch einem Tausch die Orientierung innerhalb eines Teilstücks der Route ändert, dies muss bei asymmetrischen Problemen zur Berechnung der Einsparung berücksichtigt werden. Bei symmetrischen Problemen werden lediglich die Kosten der beiden entfernten Kanten mit jenen der eingefügten verglichen.

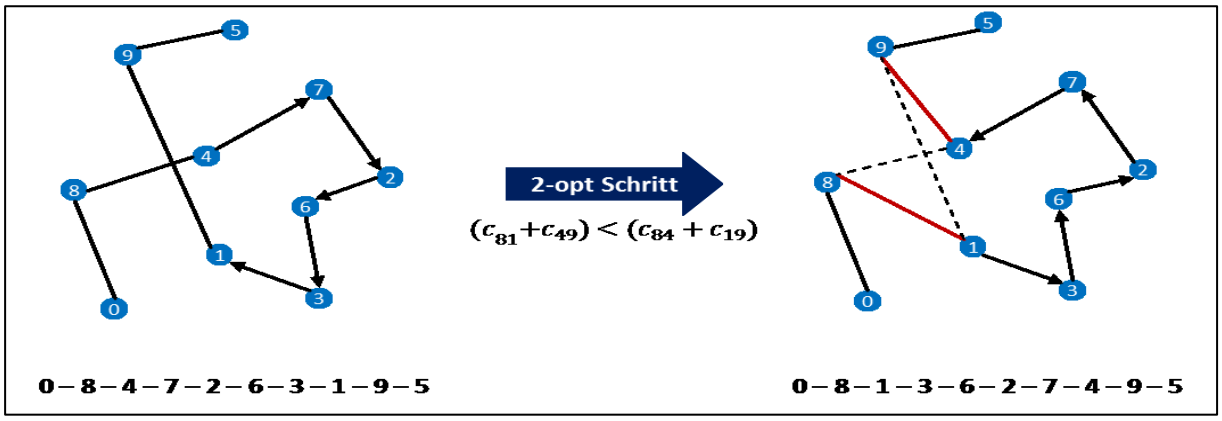


Abbildung 4: 2-opt Tausch

Bei einem 3-opt-Tausch gibt es insgesamt sieben Möglichkeiten, die es zu überprüfen gilt, wobei bei drei dieser Möglichkeiten eine Kante gleich bleibt und lediglich ein 2-opt-Schritt ausgeführt wird. Auch hier kann eines der Teilstücke der Route die Orientierung verändern, weshalb es lediglich bei symmetrischen Problemen immer ausreicht, nur die Kosten der vertauschten Kanten zu vergleichen.

Die Wahl von λ ist natürlich entscheidend für die Performance des Verfahrens, sowohl was die Rechenzeit als auch die erzielte Lösungsgüte betrifft, wobei üblicherweise ein Wert zwischen zwei und vier gewählt wird. Da durch die Erhöhung von λ die benötigte Rechenzeit exponentiell ansteigt, wird stets nach Möglichkeiten gesucht, trotz höherem λ die Rechenzeit möglichst gering zu halten. Eine einfache Möglichkeit hierfür ist zum Beispiel, für jeden Knoten nur eine bestimmte Menge an (vielversprechenden) Kanten zuzulassen. Eine andere Möglichkeit, die erzielte Lösungsgüte bei akzeptablen Rechenzeiten zu erhöhen, bietet ein variables λ, wie es bei dem **Lin-Kernighan-Verfahren** angewandt wird [28]. Zu diesem Verfahren wurden seit dessen erster Veröffentlichung im Jahr 1973 unzählige Implementierungsmöglichkeiten bzw. Abwandlungen, unter anderem in [1] und [22] (wo wiederum auf weitere Möglichkeiten verwiesen wird), beschrieben. Es zählt nach wie vor zu den effizientesten Methoden zur Lösung von TSPs, weshalb dieses nun kurz beschrieben wird:

Bei einem k-opt-Vefahren wird eine Menge an Kanten $X = \{x_1, \dots, x_k\}$ aus der Rundreise entfernt, und durch eine Kantenmenge $Y = \{y_1, \dots, y_k\}$ so ersetzt, dass wieder eine zulässige Rundreise entsteht. Bei dem Lin-Kernighan-Verfahren wird nun das k nicht vorgegeben, sondern vom Verfahren in jedem Schritt selbst bestimmt. Dies bedeutet, den beiden Kantenmengen werden immer jeweils eine neue Kante hinzugefügt, bis entweder dadurch eine bessere Rundreise gefunden wurde und der Tausch vollzogen wird, oder ein Abbruchkriterium erreicht ist. Damit dies nicht lediglich eine reine Zufallssuche darstellt, müssen natürlich passende Kriterien für die Kantenauswahl festgesetzt werden. Zunächst wird festgelegt, dass nur aufeinanderfolgende Kanten vertauscht werden können, das heißt x_i und y_i müssen sich einen Knoten teilen, sowie auch y_i und x_{i+1}. Das zweite wesentliche Kriterium ist, dass $x_i = \left(v_{x_i 1}, v_{x_i 2}\right), i \geq 2$ so gewählt werden muss, dass durch einfügen einer Kante $y_i = \left(v_{x_i 2}, v_{x_1 1}\right), i \geq 2$ wieder eine zulässige Rundreise entsteht. Dieses Kriterium gewährleistet, dass zu jeder Zeit des Verfahrens die Rundreise wieder geschlossen werden kann. Um das Verfahren zu vereinfachen (und um die Rechenzeit zu verringern), wird auch gefordert, dass die Mengen X und Y disjunkt sind, das heißt, eine Kante die entfernt wurde, darf nicht wieder eingefügt werden, und eine Kante die eingefügt wurde, darf nicht wieder entfernt werden. Nun fehlt noch ein Auswahlkriterium, das die Suche in die Richtung von vielverspre-

chenden Lösungen drängt – das Kriterium der positiven Einsparung („*The positive gain criterion*" [22]): Die Summe der Einsparung $G_i = g_1 + \cdots + g_i$ mit $g_i = c(x_i) - c(y_i)$ muss zu jedem Zeitpunkt positiv sein. Dieses Kriterium scheint auf den ersten Blick zu stark einzuschränken, da sicherlich auch negative g_i in Kauf genommen werden müssen, um eine Gesamtverbesserung zu erzielen. Doch auf Grund der Tatsache, dass es zu jeder positiven Summe von Elementen eine Reihung dieser Elemente gibt, bei der jede Partialsumme auch positiv ist (ein Beweis dazu findet sich in [28]), wird durch dieses Kriterium die Lösungsqualität nicht beeinträchtigt, sondern lediglich die Rechenzeit reduziert.

In Abbildung 5 wird ein möglicher Schritt des Lin-Kernighan-Verfahrens gezeigt, wobei die die gestrichelten Linien die entfernten Kanten darstellen, die dicken Linien die eingefügten Kanten anzeigen, und die punktierten diejenigen, die wieder zum Schließen der Tour führen würden.

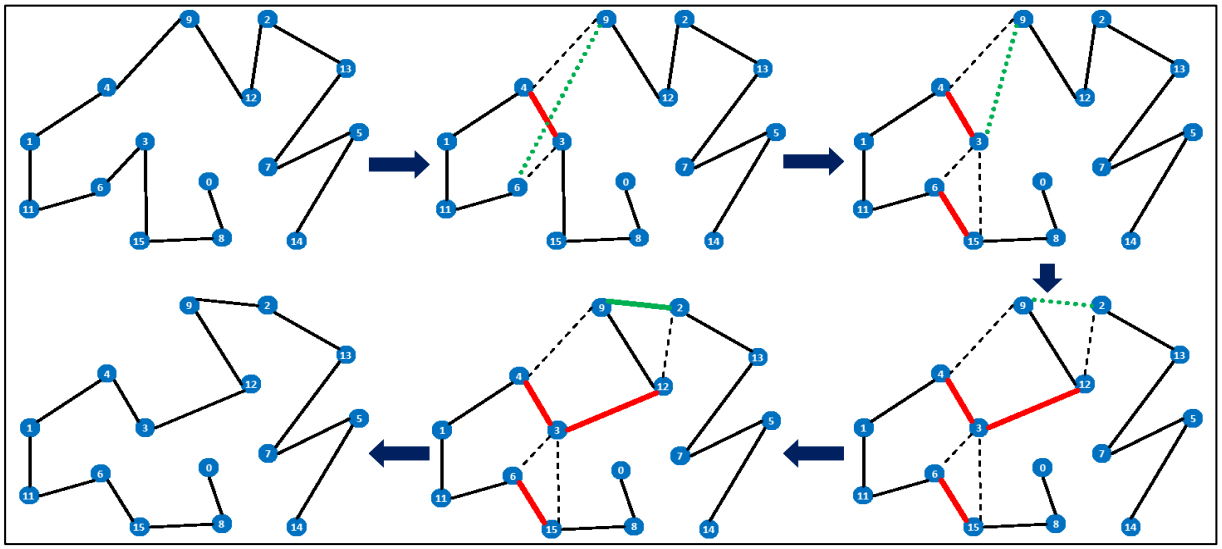


Abbildung 5: Lin-Kernighan Verfahren

Um das Verfahren noch weiter zu verbessern, können noch zusätzliche Kriterien zur Auswahl der Kanten eingeführt werden, wie zum Beispiel (bereits von Lin und Kernighan selbst vorgeschlagen) die Limitierung der möglichen Kanten auf die fünf nächsten Nachbarn zu beschränken. In [22] wird von optimalen Lösungen für 100-Städte Probleme in unter einer Sekunde (mit einem 300 MHz Prozessor) berichtet, weshalb dieses Verfahren für den Einsatz im Vehicle-Routing Problem vielversprechend scheint.

3.3 VRP-spezifische Konstruktionsheuristiken

Nachdem nun Heuristiken zum Lösen von TSPs gezeigt wurden, sollen in diesem Abschnitt Möglichkeiten präsentiert werden, wie ein gesamtes Vehicle Routing Problem gelöst werden kann. Dabei liegt das Hauptaugenmerk auf dem bis jetzt noch nicht betrachtetem Teilproblem, dem Clustering. In diesem Abschnitt werden die drei wichtigsten Konstruktionsheuristiken, die für VRPs entwickelt wurden, beschrieben. Erst im anschließenden Abschnitt 3.4 wird auf Möglichkeiten zur Optimierung unter Zuhilfenahme von Verbesserungsheuristiken bzw. Metastrategien eingegangen. Wie auch bereits bei der Beschreibung der Heuristiken für das TSP gilt auch hier, dass in den Beschreibungen von geschlossenen Touren, d.h. Touren die zum Depot zurückführen, ausgegangen wird. Auch an dieser Stelle der Hinweis, dass sich die hier angeführten Heuristiken auch auf VRPs mit offenen Touren anwenden lassen.

3.3.1 Savings Algorithmus

Der Savings-Algorithmus, erstmals veröffentlicht von G. Clarke und J.W. Wright [5], ist die am weitesten verbreitete und in der Praxis am häufigsten eingesetzte Konstruktionsheuristik für VRPs ([27],[6]). Deshalb wird auch mit der Beschreibung dieses Verfahrens begonnen, wobei auf die Ausführungen in [9] S.235ff sowie [27] zurückgegriffen wird:

Bei dem Savings-Verfahren handelt es sich um ein Simultanverfahren, da das Clustering und das Routing gleichzeitig vorgenommen werden, wobei jedoch in der Grundversion das Routing etwas vernachlässigt wird. Deshalb wird meistens am Ende des Verfahrens eine Nachoptimierung der Routen, z.B. mittels einem k-opt-Verfahren, durchgeführt. Zu Beginn des Verfahrens werden für alle n Kunden sogenannte Pendelrouten (Touren vom Depot zu dem jeweiligen Kunden und wieder zurück) erstellt. Damit entstehen n Touren $T_1, \dots, T_n$, die alle das Schema $T_i = \{v_0, v_i, v_0\}$ besitzen. An dieser Schreibweise erkennt man auch, dass es sich hierbei um ein Simultanverfahren handelt, da diese bereits die Reihenfolge der Knoten beinhaltet.

Im zweiten Schritt wird für alle Kombinationen aus zwei Kunden das „Saving", also die Einsparung, die bei Zusammenlegung der jeweiligen Pendeltouren entsteht, nach der Formel $s_{ij} = c_{i0} + c_{0j} - c_{ij}$, $i, j \in \{1, \dots, n\}, i \neq j$ errechnet. Bei symmetrischen VRPs, d.h. VRPs bei denen der zugrundeliegende Graph ungerichtet ist, gilt natürlich $s_{ij} = s_{ji}$. Die positiven Savings werden in absteigender Reihenfolge in eine Liste geschrieben. Nun gibt es zwei Möglichkeiten zur weiteren Vorgangsweise, eine sequentielle und eine parallele. Bei der sequentiellen Vorgangsweise werden die zu dem besten Saving gehörenden Pendeltouren, durch

streichen der Kanten (v_i, v_0),(v_0, v_j) und hinzufügen der Kante (v_i, v_j), zu einer Tour zusammengelegt. Diese Tour wird solange durch zusammenlegen mit einer weiteren Tour (bei der das beste noch verfügbare Saving erzielt wird) verlängert, bis es entweder kein positives Saving zu der Tour gibt, oder durch ein weiteres Zusammenlegen eine Restriktion (z.B. die maximale Laderaumkapazität) verletzt wird. Erst dann wird die nächste Tour mit dem größten noch möglichen Saving begonnen. Die parallele Vorgehensweise unterscheidet sich von dieser dahingehend, dass alle Touren gleichzeitig erweitert werden. Dies bedeutet, es wird die Savings-Liste von oben nach unten abgearbeitet, und jeder zulässige Savings-Schritt, unabhängig welche Touren er betrifft, ausgeführt. In [6] wird – nicht ganz überraschend – berichtet, dass die parallele Vorgangsweise in der Praxis meist bessere Ergebnisse erzielt.

Die wesentliche Stärke dieser Konstruktionsheuristik ist es, dass bei einer geringen Komplexität des Verfahrens, relativ gute Ergebnisse in sehr kurzer Zeit erzielt werden können. Ein weiterer Vorteil dieses Verfahrens ist es, dass durch Adaption der Berechnung der Savings-Werte die Möglichkeit besteht, zusätzliche Faktoren zu berücksichtigen. Zum Beispiel können, durch Hinzufügen einer positiven Konstanten zu dem oben gezeigten Savingswert, Fixkosten je benutztem Fahrzeug berücksichtigt werden.

3.3.2 Sweep Algorithmus

Mit dem oben beschriebenen Savings-Algorithmus wird ohne Veränderungen des Verfahrens, für jede Problemstellung immer nur ein bestimmter Tourenplan erstellt. Bei dem Sweep-Algorithmus, der den Autoren Gillet und Miller zugeschrieben wird [18], können jedoch bei einer Problemstellung mit n Kunden n verschiedene Tourenpläne generiert werden. Zu Beginn des Sweep-Algorithmus werden allen Kunden Polarkoordinaten zugeordnet, wobei der Ursprung des Koordinatensystems im Depot liegt. Anschließend werden die Kunden nach steigendem Polarwinkel sortiert. Mit dem ersten Kunden wird nun eine Tour begonnen und die nächsten Kunden in der Liste werden solange dieser Tour zugeordnet, bis eine Restriktion dies verbietet. Der erste Kunde, der nicht mehr mit der aktuellen Tour beliefert werden kann, beginnt eine neue Tour. Nach diesem Schema wird so lange vorgegangen, bis alle Kunden einer Tour zugeordnet sind. Der daraus resultierende Tourenplan ist natürlich von der Auswahl des ersten Kunden abhängig, weshalb dieser Vorgang mit einem neuen Startkunden wiederholt werden kann – dadurch entstehen die bereits erwähnten n verschiedenen Tourenpläne.

Das Routen kann bei diesem Verfahren entweder gleich bei der Tourenzusammenstellung erfolgen, was insbesondere Sinn macht, wenn es neben der Kapazitätsbeschränkung noch

z.B. zeitliche Einschränkungen gibt, oder es werden die Routen der einzelnen Touren erst am Ende festgelegt. Damit kann das Verfahren sowohl als Simultanverfahren, als auch als Sukzessivverfahren (Cluster first – route second) ausgeführt werden.

Anhand dieser kurzen Beschreibung des Verfahrens ist klar, dass die Touren bei Probleminstanzen, bei denen das Depot zentral liegt, „Blüten" (im Englischen *„petals"*) rund um das Depot bilden. Ein damit verbundener Nachteil ist, dass durch die alleinige Betrachtung des Polarwinkels keine Lösungen mit überlappenden Touren in Erwägung gezogen werden. Dies führt natürlich bei Probleminstanzen, bei denen sich die Entfernungen der Kunden zum Depot stark unterscheiden, zu einem erheblichen Nachteil gegenüber anderen Heuristiken, wie zum Beispiel dem vorher gezeigten Savings-Algorithmus. Eine weitere Schwäche dieses Verfahrens liegt darin, dass bei dem Hinzufügen der Kunden zu einer Tour nicht auf die Distanzen geachtet wird. Damit kann es dazu kommen, dass ein Kunde noch zu einer Tour gegeben wird, obwohl es günstiger wäre, mit diesem bereits eine neue Tour zu beginnen. Die möglichen Auswirkungen dieser zwei Schwächen sind in Abbildung 6 dargestellt.

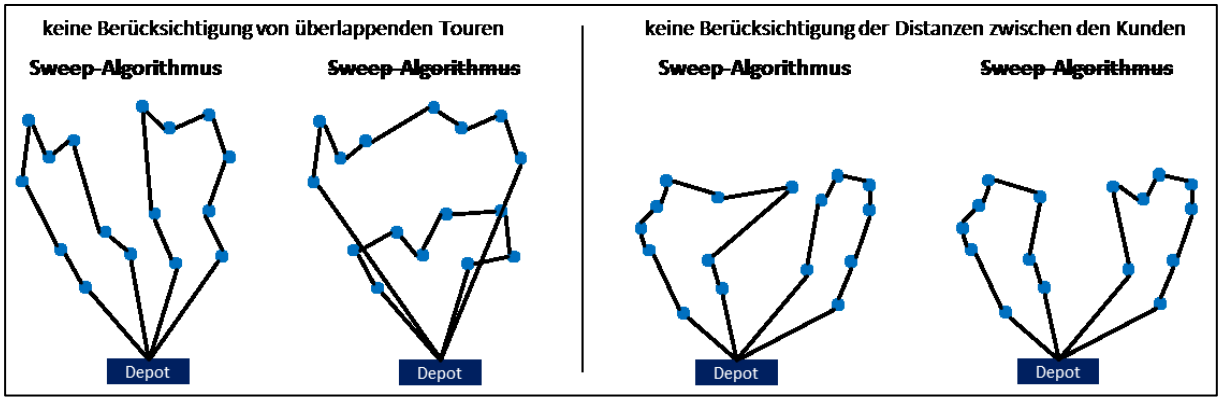


Abbildung 6: Schwächen des Sweep-Algorithmus

Aus den eben gezeigten Gründen wurden von verschiedenen Autoren von diesem Verfahren abgeleitete Heuristiken veröffentlicht (z.B. in [13], [36] und [38]), welche, in Anlehnung an die Struktur von Sweep-Lösungen, als Petal-Heuristiken bezeichnet werden. Diese werden im folgenden Abschnitt beschrieben.

3.3.3 Petal Algorithmus

Ein Petal-Algorithmus als Erweiterung des Sweep-Algorithmus wurde erstmals von A. Foster und D.M. Ryan im Jahr 1976 veröffentlicht [13]. In dieser ersten Form des Petal-Verfahrens wurde die im vorigen Abschnitt beschriebene Schwäche des Sweep-Algorithmus, dass die Distanzen zwischen den Kunden nicht bei der Bildung von Touren berücksichtigt werden, beseitigt. Dies wird durch folgende Vorgangsweise, die die Grundidee aller Petal-Algorithmen darstellt, realisiert: Anstatt lediglich die größtmöglichen Touren von aufeinanderfolgenden

Kunden innerhalb der durch die Polarkoordinaten gegebenen Reihenfolge zu erstellen, werden bei diesem Verfahren im ersten Schritt alle möglichen Touren von aufeinanderfolgenden Kunden dieser Reihenfolge erstellt. Diese Touren werden als Petals bezeichnet, die Menge aller Petals zu der gegebenen Reihenfolge als Petal-Set. Die Kosten eines Petals werden als die Kosten der dadurch definierten Route, die durch Lösen des dazugehörigen TSPs ermittelt werden, definiert. Dies bedeutet in der Regel die gesamte Distanz (vom Depot weg!) plus eventuell eine Konstante die die Fixkosten je genutztem Fahrzeug repräsentiert.

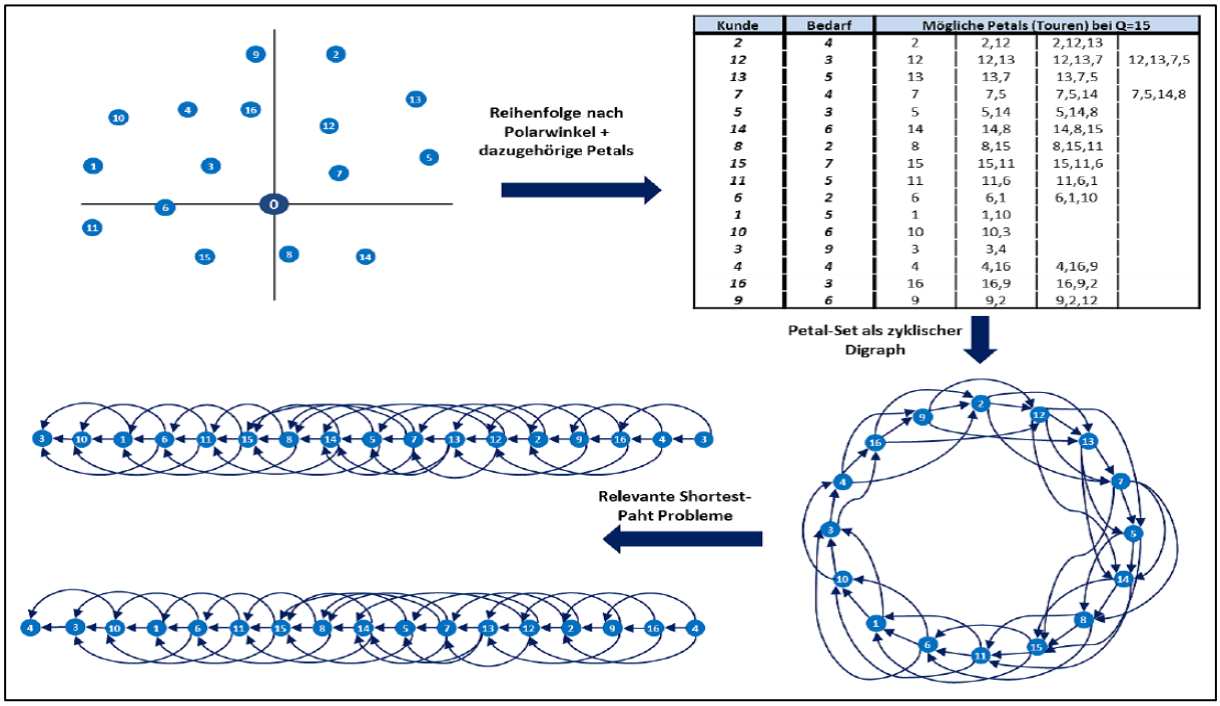
Im zweiten Schritt muss zu dem erzeugten Petal-Set das „spannende Petal-Set" (*„spanning petal set"*, [38]) mit den geringsten Kosten gesucht werden. Ein solches Set wird, angelehnt an die Nomenklatur in der Graphentheorie (siehe Abschnitt 2.1), dann als spannend bezeichnet, wenn jeder Kunde in genau einem Petal vorhanden ist. In [13] wird die optimale Lösung zu einem gegebenen Petal-Set durch das Lösen des dazugehörigen Set-Partitioning Problem (mit Hilfe einer LP-Relaxation) gefunden. Da in [38] jedoch eine weit effizientere Methode zum Auffinden des optimalen spannenden Petal-Sets gezeigt wird, wird auch an dieser Stelle lediglich diese Vorgehensweise näher betrachtet:

Das Petal-Set wird als bewerteter (zyklischer) Digraph $G = (V, E)$ dargestellt, wobei $V = \{v_1, \dots, v_n\}$ die Kunden (also Knoten) beinhaltet und $E \subseteq \{(v_i, v_j) | v_i, v_j \in V, i \neq j\}$ die Petals. Diese werden als Kanten von dem ersten Kunden des Petals, bis zu dem ersten Kunden des nächsten Petals, repräsentiert. Dabei ist die ursprüngliche Reihenfolge zu berücksichtigen, nicht jene der durch das Lösen des TSPs generierten Route. Die Bewertungen der Kanten entsprechen den Kosten der Petal-Routen. In diesem Digraphen wird nun der Kreis mit den geringsten Kosten gesucht. Aus diesem Kreis kann dann das optimale spannende Petal-Set unmittelbar abgelesen werden, da die Kanten die einzelnen Petals darstellen.

Für das Auffinden dieses kürzesten Kreises wird folgendermaßen vorgegangen: Der zyklische Digraph wird bei Knoten k aufgebrochen indem dieser dupliziert wird, wobei der Ursprungsknoten alle ausgehenden Kanten erhält und das Duplikat alle eingehenden. Zusätzlich werden alle Kanten, die am Knoten k vorbeiführen, entfernt, damit entsteht ein azyklischer Graph. Der kürzeste Pfad von k zu dessen Duplikat, welcher durch einen Shortest-Path-Algorithmus gefunden werden kann, entspricht nun dem kürzesten Kreis durch den Knoten k. Durch Lösen von n Shortest-Path Problemen (durch Aufbrechen bei den n verschiedenen Knoten) kann der optimale Kreis, und damit das optimale Petal-Set gefunden werden. In [38] wird jedoch gezeigt, dass dies nicht notwendig ist: Da alle Kanten, die am Knoten k vorbeiführen, beim Aufbrechen des Graphen bei Knoten k entfernt werden, werden damit auch alle Lösungen, die solche Kanten beinhalten, ignoriert. Könnte nun ausgeschlossen

werden, dass die optimale Lösung solch eine Kante besitzt, würde das Lösen des Shortest-Path Problems des durch k induzierten Digraphen ausreichen. Dies macht man sich nun zu Nutze, indem man zunächst den Knoten i sucht, der am wenigsten ausgehende Kanten besitzt (also jenen Kunden, bei dem am wenigsten Touren starten). Nun werden alle Digraphen untersucht, die durch Aufbrechen der Knoten, zu denen diese Kanten führen, induziert werden. Dies reicht aus, da es keine gültige Lösung gibt, die an all diesen Knoten vorbeiführt, denn dies würde dazu führen, dass mindestens einer dieser Knoten nicht in der Lösung vorhanden ist. Anders ausgedrückt: Eine gültige Lösung zur gegebenen Reihenfolge muss mindestens eine Tour enthalten, die an einem dieser Knoten startet.

In Abbildung 7 ist dieser Ablauf des Petal-Algorithmus nochmals schematisch dargestellt: Zunächst werden die Abladestellen nach aufsteigendem Polarwinkel gereiht, und alle möglichen Petals, also Touren mit aufeinanderfolgenden Kunden, generiert. Jeder dieser Petals bekommt eine Bewertung, die den Kosten der Tour entspricht (in der Abbildung aus Übersichtsgründen nicht ersichtlich). Anschließend wird daraus ein zyklischer Digraph erstellt, wobei jedes Petal als eine Kante von dem ersten Knoten des Petals, bis zum ersten Knoten, der nicht mehr zu diesem Petal gehört, dargestellt wird. Das heißt zum Beispiel die Pendeltour zum Kunden 2 wird als Kante von Knoten 2 zu Knoten 12 dargestellt, die Tour zu den Kunden 2 und 12, als Kante von Knoten 2 zu Knoten 13.



Kunde	Bedarf	Mögliche Petals (Touren) bei Q=15			
2	4	2	2,12	2,12,13	
12	3	12	12,13	12,13,7	12,13,7,5
13	5	13	13,7	13,7,5	
7	4	7	7,5	7,5,14	7,5,14,8
5	3	5	5,14	5,14,8	
14	6	14	14,8	14,8,15	
8	2	8	8,15	8,15,11	
15	7	15	15,11	15,11,6	
11	5	11	11,6	11,6,1	
6	2	6	6,1	6,1,10	
1	5	1	1,10		
10	6	10	10,3		
3	9	3	3,4		
4	4	4	4,16	4,16,9	
16	3	16	16,9	16,9,2	
9	6	9	9,2	9,2,12	

Abbildung 7: Ablauf Petal-Algorithmus

Aus dem erstellten Digraphen wird nun ein Knoten gesucht, der die geringste Anzahl an ausgehenden Kanten besitzt (z.B. Knoten 10). Nun wird dieser zyklische Graph an den Knoten,

zu denen diese Kanten führen (Konten 3 und 4), aufgebrochen. Es reicht nun aus diese zwei Shortest-Path Probleme zu lösen, denn es gibt keine gültige Lösung, die keine Tour hat, die entweder an Knoten 3 oder an Knoten 4 beginnt, denn solch eine Lösung würde zumindest den Knoten 4 sicher nicht enthalten.

Dass bei diesem Verfahren für die erzielte Lösungsgüte die ursprüngliche Reihenfolge ausschlaggebend ist, ist offensichtlich, denn es gilt die Restriktion, dass nur direkt aufeinanderfolgende Aufträge in eine gemeinsame Tour kommen können. Damit ist bei einer, auf aufsteigende Polarkoordinaten beruhenden, Anfangsreihenfolge auch klar, dass keine überlappenden oder kreuzenden Touren in Betracht gezogen werden. Aus diesem Grund wird in [38] gezeigt, dass dieses Verfahren auf jede beliebige Reihenfolge der Knoten anwendbar ist, wobei eine nicht näher beschriebene Permutation der radialen Reihenfolge vorgeschlagen wird. J.E. Beasley zeigte zum Beispiel bereits 1983 in [2] die Möglichkeit, dieses Verfahren als Route first – cluster second Verfahren auszuführen, das heißt, die Startreihenfolge durch das Lösen des TSPs für alle Kunden zu ermitteln. Einen anderen Weg, die Lösungsgüte des ursprünglichen Verfahrens zu erhöhen, wird in [36] erörtert. Anstatt die Ursprungsreihenfolge zu verändern, werden hier sogenannte *„2-petals",* also Petals die zwei Touren beinhalten, eingeführt. Dadurch entstehen auch Segmente, die überlappende oder kreuzende Touren besitzen.

3.4 Verbesserungsheuristiken und der Einsatz von Metastrategien

Nachdem im vorigen Abschnitt die wichtigsten Konstruktionsheurisitken für VRPs erörtert wurden, werden in diesem Abschnitt nun Methoden gezeigt, wie vorhandene Lösungen weiter verbessert werden können. Da in Abschnitt 3.2.2 bereits eine Möglichkeit zur Verbesserung der einzelnen Routen beschrieben wurde, wird an dieser Stelle das Hauptaugenmerk auf der Verbesserung des Clusterings liegen. Dazu wird zunächst auf die Generierung von Nachbarschaftslösungen eingegangen, und danach werden drei Metaheuristiken, die eine gute Anwendbarkeit auf VRPs versprechen, näher beschrieben. Wie bereits in Abschnitt 3.1 kurz erwähnt, stellen solche Metaheuristiken Verfahren da, die nicht auf ein bestimmtes Problem zugeschnitten sind, sondern lediglich allgemeine Vorgangsweisen zur Lösungssuche definieren. Damit können solche Verfahren durch geringe Anpassungen auch auf verschiedenste Problemstellungen angewendet werden.

3.4.1 Generierung von Nachbarschaftslösungen bei VRPs

Da für die Verbesserung einer gegebenen Lösung die Generierung von Nachbarschaftslösungen eine entscheidende Rolle spielt, werden an dieser Stelle die wichtigsten Möglichkeiten hierfür gezeigt, wofür überwiegend auf die Ausführungen in [25] zurückgegriffen wird. Es gibt grundsätzlich zwei Arten von Nachbarschaftslösungen bei VRPs: Nachbarschaftslösungen die durch Veränderung innerhalb einer Tour entstehen, und solche die durch die Veränderung des Clusterings generiert werden. Wie Nachbarschaftslösungen der ersten Art erstellt werden können, wurde bereits in Abschnitt 3.2.2 gezeigt, weshalb an dieser Stelle lediglich auf die zweite Art eingegangen wird.

Für die Generierung einer Nachbarschaftslösung, die das vorhandene Clustering verändert, gibt es im Allgemeinen drei verschiedene Möglichkeiten: eine Verschiebung, einen Austausch und eine Kreuzung. Bei einer **Verschiebung** werden ein oder mehrere zusammenhängende Knoten aus einer Tour entfernt, und in eine andere eingefügt. Ein **Austausch** wird dadurch realisiert, indem aus zwei Touren jeweils ein Knoten oder ein Teilstück der Route entfernt, und in die andere Tour eingefügt, wird. Dabei müssen die Teilstücke nicht zwingendermaßen dieselbe Anzahl an Knoten besitzen. Die letzte Möglichkeit der Veränderung, eine **Kreuzung**, ist der Austausch des letzten Teilstücks von zwei Touren. Auch hier gilt, dass die Teilstücke nicht gleich lang sein müssen. In Abbildung 8 auf der nächsten Seite sind diese drei Möglichkeiten schematisch dargestellt.

Da für ein nachbarschaftsbasierendes Suchverfahren die Nachbarschaft begrenzt werden muss, wird eine eindeutige, und möglichst einfache Definition dieser benötigt. Dafür hat Osman [30] den Begriff der *„λ-interchanges"* eingeführt, auf die viele weitere Arbeiten zurückgreifen. Hierbei steht das λ für die maximale Anzahl an Knoten, also Kunden, die in einem Schritt aus einer Tour entfernt werden können, um sie in einer anderen Tour wieder einzufügen. Dabei ist in einem Schritt lediglich der Austausch von Knoten zwischen zwei verschiedenen Touren zulässig, wodurch alle drei oben beschriebenen, und in Abbildung 8 auf der nächsten Seite gezeigten, Schritte möglich sind. Um die Anzahl an verschiedenen Nachbarschaftslösungen zu beschränken, wird in den meisten Fällen λ lediglich auf einen Wert zwischen eins und drei gesetzt. Beschrieben wird ein Schritt als Austausch $(\lambda_1, \lambda_2);\ \lambda_1, \lambda_2 \leq \lambda$, wobei λ_1 für die Anzahl an entfernten Knoten aus Tour 1 und λ_2 für die Anzahl der Koten aus Tour 2 stehen. Damit gibt es bei $\lambda = 2$ folgende Möglichkeiten: (2,2), (2,1), (1,2), (2,0), (0,2), (1,1), (1,0) und (0,1).

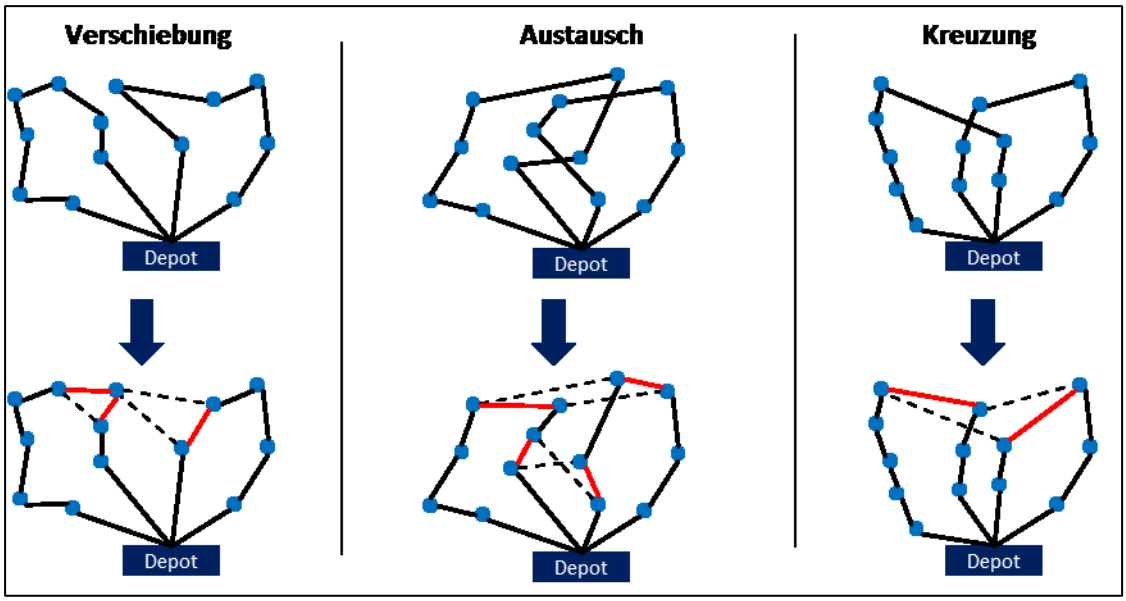


Abbildung 8: Nachbarschaftslösungen beim VRP

Da die grundlegende Vorgehensweise für eine rein lokale Suche bereits beschrieben wurde, und diese bei gegebener Nachbarschaftsdefinition ohnehin trivial ist, wird auf eine erneute Beschreibung solch eines Verfahrens verzichtet. Im nächsten Abschnitt soll dafür eine nachbarschaftsbasierende Metaheuristik gezeigt werden, die über die lokale Suche hinausgeht, und gute Lösungen für VRPs verspricht.

3.4.2 Tabu Search

Tabu Search wurde von F. Glover 1986 [19] beschrieben, und in den letzten zwei Jahrzehnten wurden zahlreiche Anwendungen dieser Methode auf das Vehicle Routing Problem publiziert. Erfolgreiche Implementierungen zur Lösung von VRPs werden unter anderem in [14], [37] und [40] beschrieben. In diesem Abschnitt wird die allgemeine Vorgangsweise dieses Verfahrens erörtert, wobei auf die Ausführungen in [7] zurückgegriffen wird.

Dieses Verfahren zählt zu den nachbarschaftsbasierenden Suchverfahren, das heißt bei dieser Methode wird in jedem Schritt nur die Nachbarschaft der derzeitigen Lösung untersucht, und die aktuelle Lösung durch die vielversprechendste Nachbarschaftslösung ausgetauscht. Danach wird mit dieser Lösung fortgefahren, bis ein Abbruchkriterium erfüllt ist. Bei solch einem Verfahren ist natürlich neben der Nachbarschaftsdefinition (siehe vorigen Abschnitt), das Auswahlverfahren innerhalb der Nachbarschaftslösungen das entscheidende Merkmal, das für die Performance des Verfahrens von Bedeutung ist. In der Regel ist natürlich die beste Lösung innerhalb der Nachbarschaft zu bevorzugen, jedoch müssen, um lokale Optima bei der Suche überwinden zu können, auch zeitweise Verschlechterungen in Kauf genommen werden. Dabei muss aber darauf geachtet werden, dass das Verfahren davor bewahrt wird „im Kreis zu gehen", also eine bestimmte Sequenz von Lösungen immer wieder zu durchlau-

fen. Um dies bewerkstelligen zu können, wird bei dem Tabu-Search-Verfahren ein Gedächtnis implementiert, das sich bereits untersuchte Lösungen merkt, und diese auf „Tabu" setzt. Dieses Verbieten von bereits untersuchten Lösungen stellt das Grundprinzip von Tabu-Search dar. Da dieses Festhalten von allen bereits untersuchten Lösungen, sowie die damit verbundene „Tabu"-Kontrolle, jedoch zu einem enormen Speicher- sowie Rechenbedarf führen, muss dieses Grundprinzip für eine erfolgreiche Implementierung des Verfahrens noch verfeinert werden. Dabei sind drei wesentliche Teilbereiche zu beachten: die Verwaltung der Tabu-Liste, oft als *„short term memory"* also Kurzzeitgedächtnis bezeichnet, die Diversifikation, die oft als Gegenstück, also *„long term memory"* bzw. Langzeitgedächtnis bezeichnet wird, sowie die Intensivierung (*„Intensification"*).

Die **Verwaltung der Tabu-Liste** stellt die Basis des Verfahrens dar und hat die bereits erwähnte Aufgabe, das Verfahren vor dem „im Kreis gehen" zu bewahren. Um dies mit einem möglichst geringen Speicherbedarf zu bewerkstelligen, hat es sich bewährt, anstatt gesamte Lösungen, nur die Schritte, die zu dieser geführt haben, zu beachten. Dies bedeutet, wenn durch Verschieben des Kunden v_i von Tour k zu Tour l die neue Lösung generiert wurde, dann wird zum Beispiel das Verschieben von v_i zu Tour k auf Tabu gesetzt. Eine andere Möglichkeit wäre es, das Entfernen von v_i von Tour l zu verbieten. Um dadurch jedoch nicht den Weg zu potenziellen guten Lösungen für die gesamte Suche zu versperren, behalten diese Züge nur für eine bestimmte Anzahl an Iterationen ihren Tabu-Status. Damit wird natürlich auch der Speicher- und Rechenbedarf begrenzt. Eine zusätzliche Möglichkeit, vielversprechende Gebiete im Lösungsraum nicht unnötigerweise zu versperren, ist die Einführung von sogenannten Aspirationskriterien. Erfüllt ein Schritt ein solches Kriterium, so kann er auch durchgeführt werden, wenn er auf Tabu gesetzt ist. Ein häufig eingesetztes Aspirationskriterium ist das Kriterium der besten Lösung, das heißt, ist die neue Lösung besser als die beste bisher gefundene Lösung, so kann der Schritt trotz Tabu-Status vollzogen werden.

Der Teilbereich **Diversifikation** beinhaltet alle Maßnahmen, die dafür sorgen, dass der Lösungsraum möglichst großflächig abgesucht wird, um dadurch vielversprechende Regionen zu finden. Eine Möglichkeit hierfür wurde bereits von Glover in [19] gezeigt und von Taillard in [40] weiterentwickelt: Jedem Schritt werden zusätzlich zu den tatsächlichen Kosten, also den Kosten die durch die Veränderung der Touren verursacht werden, zusätzliche Kosten zugeordnet. Diese sind abhängig von der Häufigkeit, in der der jeweilige Schritt bereits durchgeführt wurde (deshalb wird dieser Teilbereich auch oft als Langzeitgedächtnis bezeichnet). Damit wird das Verfahren dazu gedrängt, noch nicht untersuchte Gebiete des Lösungsraums zu bevorzugen.

Die **Intensivierung** ist, als Gegenstück zur Diversifikation, für die genaue Untersuchung von vielversprechenden Regionen verantwortlich. Dies bedeutet, dass ein weiterer Speicher eingeführt wird, der die besten Lösungen festhält. Auf die hier gespeicherten Lösungen wird, meist lediglich am Ende des Tabu-Search-Verfahrens, ein weiteres Verfahren zur Verbesserung der Lösung angewandt. Eine Möglichkeit diese Bereiche intensiver zu untersuchen, ist zum Beispiel die zu untersuchende Nachbarschaft zu erweitern, d.h. die maximale Anzahl an zu vertauschenden Kunden zu erhöhen. Eine andere mögliche Verwendung dieses Speichers ist es, die hier gespeicherten Lösungen einem populationsbasierenden Verbesserungsverfahren, wie dem im folgenden Abschnitt beschriebenen genetischen Algorithmus, als Startlösungen zur Verfügung zu stellen.

3.4.3 Genetischer Algorithmus

Die Idee zum genetischen Algorithmus wurde erstmals von J. Holland 1975 [23] aufgegriffen, und insbesondere in den letzten zwei Jahrzehnten erschienen zahlreiche Artikel, die Anwendungs- und Implementierungsmöglichkeiten im Bereich der Optimierung zeigen (einen Literaturüberblick bezüglich Implementierungen für VRPs bietet [16]).

Der Grundgedanke dieses Verfahren orientiert sich an dem natürlichen Evolutionsprozess: Lösungen für das vorliegende Problem werden als Individuen innerhalb einer Population beschrieben. Aus dieser Population wird eine bestimmte Anzahl an Individuen ausgewählt und gekreuzt, das heißt, die Eigenschaften der Individuen werden kombiniert. Bei dieser Kreuzung wird zusätzlich die Möglichkeit von Mutationen, also zufälligen Veränderungen von Eigenschaften, in das Verfahren integriert. Durch diesen Vorgang entsteht eine neue Generation an Individuen (Lösungen), mit der dieser Vorgang wiederholt werden kann. Damit dieser iterative Prozess in die Richtung von guten Lösungen tendiert, werden bei der Auswahl der Individuen, sowie bei der Kreuzung gute Lösungen bzw. Teile von Lösungen bevorzugt (*„survival of the fittest"*). Im Folgenden werden die wichtigsten Teilaspekte genetischer Algorithmen näher beschrieben, wobei lediglich die allgemeine Vorgangsweise, und keine detaillierte Implementierungsmöglichkeit, gezeigt wird.

Bevor diese Heuristik auf ein Problem angewendet werden kann, muss zunächst eine geeignete **Codierung** eines Individuums, das heißt einer Lösung des Problems, ausgewählt werden. Da für kombinatorische Optimierungsprobleme eine Darstellung als Bitmuster, das heißt als Aneinanderreihung von Binärvariablen, meist nicht in Frage kommt, erfolgt hierfür in den meisten Fällen die Darstellung direkt als Permutation. Dies bedeutet die Codierung eines Individuums hat die Form „2-5-1-4-3" und bedeutet z.B. dass die Kunden in dieser Reihen-

folge bedient werden. Zusätzlich zu der Darstellung der Lösung wird noch ein Fitness-Wert für jedes Individuum benötigt, der die Qualität der dargestellten Lösung widerspiegelt.

Zu Beginn des Verfahrens wird eine **Ausgangspopulation** benötigt, die meist durch einfache Konstruktionsheuristiken erzeugt wird. Da bei diesem Verfahren neue Lösungen hauptsächlich durch Kombination bereits bekannter Lösungen generiert werden, ist es natürlich von Vorteil, wenn sich diese Individuen einerseits stark unterscheiden (um eine „genetische Vielfalt" zu gewährleisten), aber andererseits auch bereits gute Lösungen (und somit gutes „Genmaterial") vorhanden sind. Die Größe der Ausgangspopulation, bzw. der Populationen allgemein, muss an die jeweilige Problemstellung und an die verwendeten Operatoren der Selektion bzw. Kreuzung angepasst werden.

Nachdem eine Codierung festgelegt wurde, und eine Ausgangspopulation vorhanden ist, kann mit dem ersten Schritt des eigentlichen Verfahrens, der **Selektion**, begonnen werden. In diesem Schritt werden diejenigen Individuen aus der Population ausgewählt, die in den „Paarungspool" kommen, und somit der Kreuzung zugeführt werden. Dabei wird durch Bevorzugung von Individuen mit besseren Fitness-Werten ein gewisser „Selektionsdruck" auf das Verfahren ausgeübt, und somit das Verfahren zur Konvergenz gegen gute Lösungen getrieben. Um jedoch nicht vorzeitig in ein lokales Optimum getrieben zu werden, muss eine Balance zwischen diesem Selektionsdruck und dem Aufrechterhalten von genetischer Vielfalt gefunden werden. Auch wenn die Selektion entscheidend für die Qualität einer Implementierung dieses Verfahrens ist, wird an dieser Stelle nicht näher auf die zahlreichen Möglichkeiten der Auswahlverfahren im Einzelnen eingegangen, da all diese Verfahren ohnehin dasselbe Muster besitzend: Die stärksten Individuen werden zwar bevorzugt (um das Verfahren in Richtung guter Lösungen zu drängen), doch bekommen auch die etwas schwächeren Individuen eine Chance in den Paarungspool zu kommen (um einer zu schnellen Konvergenz gegen ein lokales Optimum entgegenzuwirken).

Bei der **Kreuzung** geht es nun darum, die Merkmale der ausgewählten Individuen zu kombinieren, um dadurch neue Individuen, und somit eine neue Generation, zu schaffen. Dieser Vorgang erfolgt bei Problemstellungen, bei denen die Individuen in Binärmuster codiert sind, meist folgendermaßen: Es werden zwei „Eltern" aus dem Paarungspool gewählt und das Muster der Individuen in zwei oder mehrere Teile geteilt. Anschließend werden die Nachkommen durch zusammenfügen der Teile von Vater und Mutter erzeugt. Da bei kombinatorischen Optimierungsproblemen in der Regel jeder Wert nur einmal vorkommen darf, muss dieses Kreuzungsverfahren bei Reihenfolgeproblemen etwas verändert werden: Als erstes wird das zu vertauschende Teilstück gewählt. Danach werden die Eltern kopiert und jeweils

die Elemente entfernt, die vom anderen Elternteil aus dem gewählten Teilstück eingefügt werden sollen. Zum Schluss werden die restlichen Elemente dermaßen verschoben, dass das gewählte Teilstück frei ist, und der Tausch vollzogen werden kann. Die eben beschriebene Kreuzungsmethoden sind in Abbildung 9 nochmals dargestellt.

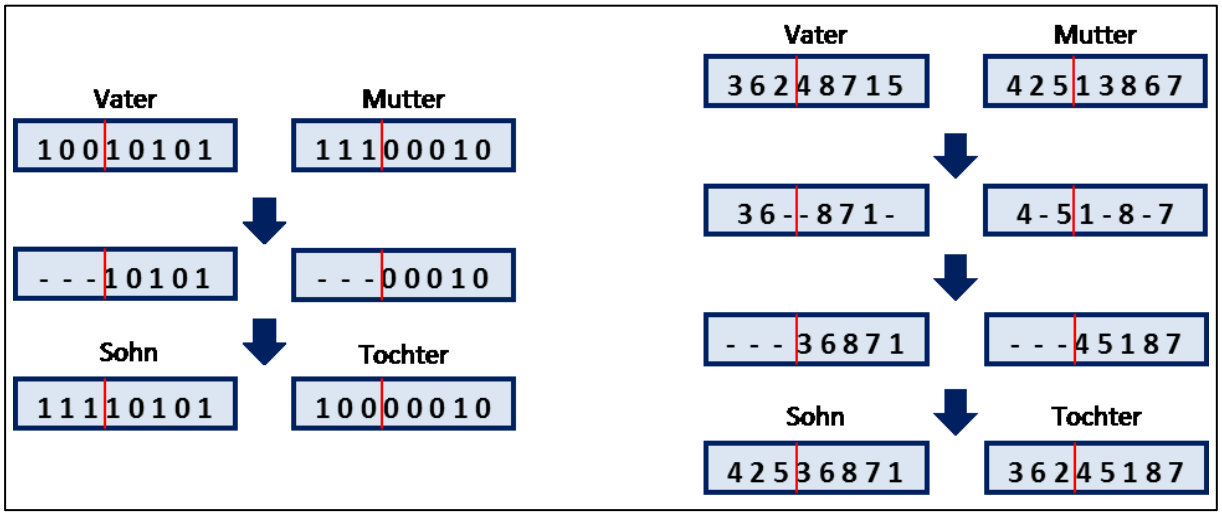


Abbildung 9: genetischer Algorithmus - Kreuzungsverfahren

Wie bereits kurz erwähnt, kann nach erfolgter Kreuzung noch eine **Mutation** durchgeführt werden. Dabei wird die Struktur der Nachkommen zufällig geändert. Dieser Schritt wird deshalb durchgeführt, da bei der Kreuzung nur bekannte Strukturen neu kombiniert, nicht aber neue Strukturen geschaffen werden. Um dem Algorithmus jedoch das Konvergieren gegen gute Lösungen nicht vollständig zu nehmen, darf dieser Schritt lediglich im geringen Ausmaß durchgeführt werden. Eine Mutation kann zum Beispiel bei einem Bitmuster durch „Umschalten" einzelner Bits erfolgen, oder bei einer Reihenfolgedarstellung durch Vertauschen zweier Elemente.

Nach Generierung der neuen Generation muss für die eben geschaffenen Individuen noch eine **Bewertung** durchgeführt werden, ehe wieder mit der Selektion begonnen werden kann. Dies bedeutet, es wird den einzelnen Individuen ein Fitness-Wert zugeordnet, der den Zielerreichungsgrad widerspiegelt. Insbesondere bei Vehicle Routing Problems kann dieser Schritt sehr rechenintensiv sein und den Hauptteil der gesamten Rechenzeit beanspruchen. Dabei hängt die Rechenzeit vor allem von der Art der Codierung und der Genauigkeit, mit der die Bewertung erfolgen soll, ab.

Die hier beschriebene Vorgangsweise zeigt natürlich lediglich das Grundmodell des genetischen Algorithmus, und kann (bzw. sollte) weiter angepasst und erweitert werden. Eine häufig angewandte Anpassung wäre zum Beispiel, dass nicht bei jeder Iteration die gesamte Population ausgewechselt wird, sondern lediglich ein (schwacher) Teil. Für detailliertere Be-

schreibungen und Implementierungsmöglichkeiten dieser Grundidee sei zum Beispiel auf [29] verwiesen.

Bevor nun die Anwendbarkeit dieser Vorgangsweise näher betrachtet wird, bzw. ein Vergleich mit den anderen Algorithmen erfolgt, wird im nächsten Abschnitt noch eine weitere Heuristik, die ebenfalls ihren Ursprung in Naturbeobachtungen hat, beschrieben.

3.4.4 Ameisenalgorithmus

Die Idee zum Ameisenalgorithmus wurde erstmals 1996 von M. Dorigo, V. Maniezzo und A. Colorni [11] unter der Bezeichnung *„Ant System (AS)"* publiziert, und zählt somit zu den jüngsten (erfolgreichen) Metaheuristiken, die auf kombinatorische Optimierungsprobleme angewandt werden. Bereits ein Jahr danach wurde von M. Dorigo und L.M. Gambardella eine Weiterentwicklung dieser Heuristik unter der Bezeichnung *„Ant Colony System (ACS)"* veröffentlicht [10], die wesentliche Verbesserungen mit sich brachte. Aus diesem Grund wird in dieser Arbeit gleich auf diese Weiterentwicklung eingegangen.

Den Ursprung hat diese Heuristik, wie der Name bereits vermuten lässt, in der Beobachtung von Ameisenkolonien. Bei einem Experiment, bei dem es zwischen einer Ameisenkolonie und einer Futterquelle einen langen und einen kurzen Weg gab, wurde beobachtet, dass zunächst beiden Wege gleichermaßen stark frequentiert werden, jedoch nach einer gewissen Zeit (fast) alle Ameisen nur noch den kürzeren Weg benutzen. Dies kann folgendermaßen erklärt werden: Während eine Ameise auf Futtersuche ist, gibt diese Pheromone ab, die nach einer gewissen Zeit wieder verdunsten. Kommt sie an eine Gabelung, so wählt diese grundsätzlich zufällig einen Weg aus, jedoch lässt sie sich dabei von den abgegebenen Pheromonen bereits vorangegangener Ameisen beeinflussen – sie bevorzugt Wege mit höherer Pheromonkonzentration. Da nun eine Ameise, die den kürzeren Weg gewählt hat, schneller wieder zu der Gabelung zurückkehrt (und natürlich auch am Rückweg Pheromone abgibt), steigt auf dem kürzeren Weg die Pheromonkonzentration schneller an. Damit werden die nachfolgenden Ameisen vermehrt diesen Weg wählen, wodurch der Unterschied natürlich immer weiter erhöht wird, bis (fast) alle Ameisen auf Grund der erhöhten Pheromonkonzentration den kürzeren Weg wählen. Diese Tendenz zum kürzeren Weg wird natürlich auch durch die Verdunstung positiv beeinflusst.

Um mit Hilfe dieser Vorgehensweise kombinatorische Optimierungsprobleme wie das TSP oder das VRP effizient lösen zu können, muss dieses Verfahren aus der Natur noch ein wenig angepasst werden. Auf Grund des Ursprungs dieser Idee liegt es nahe, dies anhand des Travelling-Salesman Problem zu beschreiben (wie auch in [10]):

Zunächst wird festgelegt, dass die Ameisen erst am Rückweg Pheromone abgeben, und dies abhängig vom Erfolg, also von der Länge des gefundenen Weges. Eventuell vorhandene Kreise müssen natürlich zuvor entfernt, bzw. dürfen erst gar nicht ermöglicht, werden. Weiters wird neben der Pheromonkonzentration, die Länge der einzelnen Kanten als zweites Entscheidungskriterium bei der Wahl des folgenden Knotens berücksichtigt. Dadurch können die Wahrscheinlichkeiten ausgehend von Knoten i den Knoten j zu besuchen, nach folgender Formel berechnet werden:

$$p_{ij} = \frac{(\tau_{ij})^{\alpha}(\eta_{ij})^{\beta}}{\sum_{j \in U_{k,i}} (\tau_{ij})^{\alpha}(\eta_{ij})^{\beta}} \quad [3.1]$$

Dabei gibt τ_{ij} die Pheromonkonzentration der Kante(v_i, v_j) an, und η_{ij} stellt die Bewertung der Kante (v_i, v_j) dar, wobei für TSPs in der Regel der Kehrwert der Kantenlänge genommen wird. In [35] wird für VRPs eine Variante gezeigt, bei der anstatt der Kantenlänge, der Savings-Wert herangezogen wird (siehe Abschnitt 3.3.1). Durch die Parameter α und β können die zwei Auswahlkriterien gewichtet werden. Die Menge $U_{k,i}$ enthält alle Knoten in der Umgebung von i, die für die Ameise k noch zulässig sind, also noch nicht besucht wurden. Bei dem ACS erfolgt die Auswahl des folgenden Knotens nun nach der pseudozufälligen Proportionalitätsregel (*„pseudo-random-proportional rule*", [10]): Es wird ein q zufällig aus dem Intervall [0,1] ausgewählt, und mit einem vordefinierten q_0 verglichen. Ist $q < q_0$, so wird der folgende Knoten j als $max_{j \in U_{k,i}}\{(\tau_{ij})^{\alpha}(\eta_{ij})^{\beta}\}$ bestimmt, ansonsten wird der nächste Knoten auf Basis der Wahrscheinlichkeiten aus [3.1] gewählt. Dies bedeutet, je höher q_0, desto stärker wird das Verfahren zur Konvergenz gegen die bisher meist frequentierte Route getrieben.

Die Pheromonkonzentrationen werden bei dem in [10] beschriebenen ACS nach zwei Regeln verändert, global und lokal. Das globale Pheromon-Update erfolgt nach jeder Iteration aller m Ameisen nach folgender Formel:

$$\tau_{ij} \leftarrow (1-\rho)\tau_{ij} + \rho * \Delta\tau_{ij} \quad [3.2]$$

Dabei steht ρ für die Verdunstungsrate und $\Delta\tau_{ij}$ ist die Pheromonerhöhung, die für alle Kanten der bisher besten Route den Wert $\frac{1}{c(T_{best})}$ annimmt, und für allen anderen Kanten 0 ist ($c(T_{best})$ steht für die Kosten der bisher besten Route). Dies bedeutet, die globale Pheromonveränderung übernimmt einerseits das Verdunsten des Pheromons auf allen Kanten, und andererseits drängt sie das Verfahren in die Richtung der bisher besten Lösung.

Das lokale Pheromon-Update wird von jeder Ameise durchgeführt, sobald diese eine Kante nutzt. Dieses Update erfolgt nach der Formel [3.3].

$$\tau_{ij} \leftarrow (1-\lambda)\tau_{ij} + \lambda * \tau_0 \qquad [3.3]$$

In dieser Formel ist λ ein Faktor zwischen 0 und 1, und τ_0 gibt den Initialwert der Pheromonkonzentrationen an, für den ein Wert von $\frac{1}{n*c(T_{nN})}$ vorgeschlagen wird (n entspricht der Anzahl an Knoten, $c(T_{nN})$ steht für die Kosten der Route, die durch das Nearest-Neigbor-Verfahren ermittelt wurde).

Die Performance von Implementierungen solcher Ameisenalgorithmen hängt natürlich in erster Linie von dem „Pheromon-Management" ab, denn die Suche wird überwiegend durch die verschiedenen Pheromonkonzentrationen gesteuert. Dabei muss, wie bei allen Metaheuristiken, eine Balance zwischen der Exploration des gesamten Lösungsraums und einer Konvergenz gegen gute Lösungen gefunden werden. Neben der eben gezeigten Vorgehensweise bezüglich des Pheromon-Updates, werden in der Literatur auch noch andere Möglichkeiten gezeigt. Die wichtigsten davon sind das elitäre Ameisensystem, bei dem die beste bisher gefundene Route stärker markiert wird, die rangbasierende, bei der das Pheromon-Update abhängig vom Rang innerhalb der Iteration erfolgt, und das Max-Min Ameisensystem, bei dem die Pheromonkonzentrationen nach oben und unten beschränkt sind. Einen Überblick über die verschiedenen Implementierungsmöglichkeiten, und eine generell gute und umfangreiche Beschreibung von Ameisenalgorithmen, gibt [12] (die verschiedenen Möglichkeiten des Pheromon-Updates werden von Seite 72 bis Seite 82 beschrieben).

3.5 Resümee über die publizierten Ergebnisse

Nachdem in diesem Kapitel nun verschiedene Lösungsansätze für Vehicle Routing Probleme erörtert wurden, soll an dieser Stelle ein kurzer Vergleich zwischen diesen angestellt werden. In diesem Vergleich wird versucht, auf Basis der publizierten Ergebnisse innerhalb der verwendeten Literatur, herauszufinden, welche Stärken und Schwächen die einzelnen Verfahren aufweisen. Daraus sollten anschließend Schlüsse gezogen werden können, welche Verfahren für welche Problemstellungen geeignet erscheinen. Da bei realen Problemstellungen der Einsatz von Verbesserungsheuristiken sicherlich nötig ist, und damit die Wahl der Konstruktionsheuristik kaum Auswirkungen zeigt, sollen lediglich die drei vorgestellten Metaheuristiken verglichen werden:

Bei Durchsicht der vorhandenen Literatur fällt zunächst auf, dass es zu Tabu Search Implementierungen sicherlich am meisten Publikationen gibt. Diese werden auch fast immer als Benchmark für andere Heuristiken herangezogen, wobei die Implementierungen von [37] und [40] anscheinend die besten Ergebnisse für die gängigsten Probleminstanzen, die von N. Christofides, A. Mingozzi und P. Toth stammen [4], erzielen. Bei vielen vergleichenden Studien, wie zum Beispiel [6], werden auch häufig nur Tabu-Search-Implementierungen herangezogen. In [15] werden viele verschiedene Heuristiken bzw. Metaheuristiken miteinander verglichen, und auch hier kommen die Autoren zu dem Schluss, dass Tabu Search Algorithmen die übrigen Strategien dominieren.

Dies lässt zunächst vermuten, dass die zwei populationsbasierenden Metaheuristiken eindeutig unterlegen sind, und man lediglich eine Tabu Search Implementierung in Betracht ziehen sollte. Doch bei genauerer Betrachtung kann erkannt werden, dass dies für reale Problemstellungen nicht so eindeutig ist und sowohl genetische, als auch Ameisen-Algorithmen, ihre Vorteile haben. Wie bereits erwähnt, werden zwar mit der Tabu-Suche für die oben erwähnten Probleminstanzen die besten Lösungen erzielt, diese stammen aber aus dem Jahr 1979 und haben maximal 200 Abladestellen (und sind damit in der Regel kleiner als reale Problemstellungen). Bei neueren, größeren Test-Instanzen kann jedoch keine Dominanz von Tabu-Search-Implementierungen festgestellt werden, eher umgekehrt. Für die Problem-Instanzen von [21] (200 bis 480 Kunden), die neben jenen von [24] (200 bis 1000 Kunden) die am häufigsten verwendeten sind, wurden die besten Ergebnisse mit einem Genetischen Algorithmus erzielt [33]. Aber auch Ameisenalgorithmen liefern für diese größeren Problemstellungen gute Ergebnisse (z.B. die Implementierung von [34]). Als eines der erfolgreichsten Tabu-Search-Implementierungen für solche VRP-Instanzen sei [43] zu erwähnen. Einen sehr aktuellen und umfangreichen Überblick über die derzeitigen Entwicklungen im Bereich der „*large scale VRPs*" gibt [17].

Zusammenfassend kann gesagt werden, dass für größere Probleminstanzen immer stärker hybride Algorithmen zum Einsatz kommen. Dies kann dadurch erklärt werden, dass die verschiedenen Heuristiken ihre unterschiedlichen Stärken haben. So kann mit Hilfe von populationsbasierenden Heuristiken ein großer Lösungsraum in relativ kurzer Zeit großflächig „gescannt" und damit vielversprechende Gebiete ausfindig gemacht werden. Nachbarschaftsbasierende Heuristiken wie Tabu Search können sich auf Grund ihrer Grundstruktur natürlich nicht so schnell im Lösungsraum bewegen. Dafür sind sie für die intensivere Untersuchung von vielversprechenden Teilgebieten sicherlich besser geeignet. Auch rein lokale Suchverfah-

ren sind in den meisten Implementierungen zu finden. Diese sollen den Lösungen den letzten „Feinschliff" verpassen.

Neben dem Trend zu hybriden Algorithmen, kann auch ein zweiter Trend erkannt werden, der Trend zu parallelen Suchverfahren. Dieser Trend wird sich auf Grund des technischen Fortschritts - Multi-Core-Prozessoren werden in Zukunft sicherlich zum Standard - auch durchsetzen. Dies bedeutet, neue Implementierungen, die auch in der Praxis eingesetzt werden sollen, werden auf eine (zumindest teilweise) parallele Verarbeitung nicht verzichten können. Insbesondere deshalb, weil in der Praxis neben der Lösungsqualität, die Rechenzeit eine wesentliche Rolle spielt.

4 Vorliegende Problemstellung

In den vorigen Kapiteln wurde das Vehicle Routing Problem näher erörtert und gezeigt, wie solche Probleme gelöst werden können. An dieser Stelle soll nun die tatsächlich vorliegende Problemstellung, die den Anstoß zu dieser Arbeit gegeben hat, beschrieben werden. Dabei stehen zwei Aspekte im Vordergrund: Zunächst soll anhand dieser Ausführungen gezeigt werden, welche verschiedenen Faktoren innerhalb dieses Unternehmens im Bereich der Tourenplanung zu beachten sind. Des Weiteren wird in diesem Kapitel das gesamte Konzept zur Verbesserung der Tourenplanung in diesem Unternehmen kurz erörtert. Dabei wird insbesondere auf jene Teilbereiche eingegangen, die im engen Zusammenhang mit der Tourenzusammenstellung, und somit dem VRP, stehen.

4.1 Das Unternehmen und dessen Tätigkeitsfeld

Diese Arbeit wird innerhalb eines Projekts zur Optimierung der Tourenplanung bei der ADA Möbelfabrik GmbH, bzw. deren Mutterkonzern Elefant Holding AG, verfasst. Die ADA Möbelfabrik GmbH mit Sitz in Anger bei Weiz produziert Polstermöbel sowie Betten (inklusive Matratzen und Lattenroste) für das höhere Preissegment. Diese Produkte werden ausschließlich über den Möbelhandel vertrieben, wobei der überwiegende Teil in den Ländern Österreich, Deutschland und der Schweiz abgesetzt wird. Es werden jedoch auch Möbelhändler in anderen europäischen Ländern, insbesondere in Italien, Tschechien, Slowenien, der Slowakei sowie den Benelux-Staaten, beliefert. Neben dem Werk in Anger gibt es noch zwei weitere Tochtergesellschaften in Ungarn (in Nova und in Körmend), sowie eine in Rumänien (Salonta). Diese Werke bedienen eher das niedrige bis mittlere Preissegment, wobei in Körmend und Salonta lediglich Polstermöbel produziert werden, in Nova nur Matratzen und Lattenroste. Die belieferten Länder (und auch Kunden bzw. Möbelhäuser) sind für alle vier Werke überwiegend dieselben.

Das Unternehmen entstammt einer kleinen Seilerei die um 1900 in Anger gegründet wurde. Diese sattelte um 1950 auf eine Matratzenproduktion um, die bis zum Jahr 1970 zu einer Möbelfabrik mit ca. 300 Mitarbeitern anwuchs. Anfang der 90er Jahre gründete die Elefant Holding AG die erste ungarische Tochtergesellschaft in Körmend. Dieses Werk wuchs sehr schnell, und so wurde Ende der 90er die zweite ungarische Tochtergesellschaft in Nova gegründet. 2005 folgte mit dem Werk in Salonta schließlich die Expansion in Richtung Rumäni-

en. Heute ist ADA der größte Möbelhersteller Österreichs und beschäftigt derzeit insgesamt rund 2.000 Mitarbeiter.

4.2 Die Tourenplanung innerhalb des Unternehmens

In diesem Abschnitt sollen die wichtigsten Erkenntnisse, die aus der Ist-Analyse, die im Rahmen dieses Projekts durchgeführt wurde, gewonnen wurden, kurz zusammengefasst werden. Die Tourenplanung stellt in diesem Unternehmen eine wichtige planende sowie steuernde Tätigkeit dar, die weit über die alleinige Tourenzusammenstellung hinausgeht. Bevor nun auf die wichtigsten Aufgaben dieser Tourenplanung eingegangen wird, sollen zunächst die für die Tourenplanung relevanten Stationen, die ein Kundenauftrag im Unternehmen durchläuft, kurz beschrieben werden. Dadurch wird einerseits das Verständnis der Problemstellung erleichtert und zweitens wird so die Eingliederung der Tourenplanung in den gesamten Geschäftsprozess ersichtlich.

4.2.1 Stationen des Kundenauftrags

In Abbildung 10 sind die wichtigsten Stationen eines Kundenauftrags vom Auftragseingang bis zur Auslieferung gezeigt.

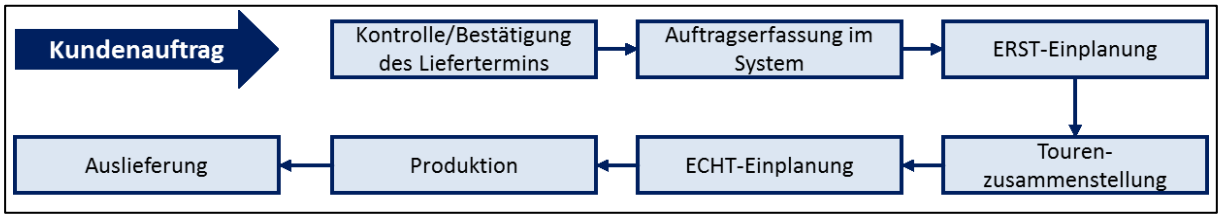


Abbildung 10: Stationen des Kundenauftrags

Nachdem der Kundenauftrag im Unternehmen ankommt, wird diesem eine Lieferwoche zugeordnet, die dann auch dem Kunden mitgeteilt wird. Hierbei wird natürlich versucht, den Wunschtermin des Kunden erfüllen zu können, jedoch wird zu diesem Zeitpunkt auch bereits darauf geachtet, ob es eine Tour in das Gebiet dieses Kunden in dem gewünschten Zeitraum gibt. Dies ist insbesondere bei Kunden in Gebieten mit einer geringeren Auftragsdichte relevant. Zusätzlich wird auch bereits bei der Zusage einer Lieferwoche versucht, auf die Auslastung in der Produktion Rücksicht zu nehmen – dies ist aber derzeit kaum realisierbar.

Wurde dem Kunden eine Lieferwoche zugesagt, wird der Auftrag im System erfasst, wobei als Liefertermin der Montag der zugesagten Woche eingetragen wird. Auf Basis dieser Erfassung wird der Auftrag eingeplant (ERST), dies bedeutet die notwendigen Materialien und

Arbeitsgänge werden reserviert. Da hier alle Aufträge mit einer ganzen Woche eingeplant werden, und der genaue Liefertermin innerhalb dieser Woche zu diesem Zeitpunkt noch nicht feststeht, wird dadurch lediglich ein Wochenüberblick über die Bedarfe geschaffen.

Die tatsächliche Tourenzusammenstellung erfolgt in der Regel zwei Wochen vor der Auslieferung. Hier werden die einzelnen LKW-Touren festgelegt, wobei, auf Grund der unzureichenden Unterstützung durch eine bedarfsorientierte Informationsbereitstellung, diese Einteilung bereits bei der Auftragserfassung begonnen wird (siehe Abschnitt 4.2.3). Bei der Festlegung der Touren steht natürlich das grundlegende Ziel „die Aufträge zum vom Kunden gewünschten Termin zu den geringstmöglichen Kosten auszuliefern" im Vordergrund. Zusätzlich muss (oder sollte) jedoch auch auf die Engpasskapazitäten innerhalb der Produktion Rücksicht genommen werden.

Mit der Erfassung der Touren im System wird meist ca. acht bis zehn Werktage vor der Auslieferung gewartet, um damit noch leichter auf kurzfristige Aufträge reagieren zu können. Bis spätestens acht Werktage vor der Auslieferung muss die Tour jedoch erfasst werden, da dies der minimale Zeitraum ist, den die Produktionsplanung bzw. die Produktion benötigen. Jedoch auch innerhalb dieser acht Tage kann es noch zur Einplanung von kurzfristigen Aufträgen kommen. Diese betreffen insbesondere Matratzen, da für diese wesentlich kürzere Durchlaufzeiten als für Polstermöbel gefordert werden. Solche kurzfristigen Aufträge werden dann einer geeigneten existierenden Tour zugeteilt, wobei bei vielen Touren bewusst Laderaum für Matratzen freigelassen wird.

Nach der Erfassung der Touren im ERP-System kommt es zu einer neuerlichen Einplanung der Aufträge (ECHT) – diese stellt die Grundlage für die Produktionsplanung dar. Nach der Produktion werden die Produkte in die einzelnen LKWs geladen. Dabei kommt es gelegentlich vor, dass Produkte nicht mit der geplanten Tour mitkommen. Die Hauptgründe dafür sind Verspätungen innerhalb der Produktion, Stock-Out von benötigten Rohstoffen (vor allem bestimmter Stoffe), oder „Überbuchung" des Fahrzeugs, also wenn der Laderaum des Fahrzeugs bei der Tourenplanung überschätzt wurde. Solche Produkte müssen dann möglichst rasch als Nachlieferung zum Kunden gebracht werden.

4.2.2 Aufgaben und Herausforderungen

Wie bereits kurz erwähnt, gehen die Aufgaben der Tourenplanung über die reine Tourenzusammenstellung hinaus. Da sich jedoch dieses Projekt, und damit auch diese Arbeit, überwiegend mit dieser Kernaufgabe der Tourenplanung beschäftigt, sollen an dieser Stelle lediglich die damit in unmittelbaren Zusammenhang stehenden Aufgaben und Herausforderungen

beschrieben werden. Als übergeordnetes Ziel dieser Kernaufgabe steht die rechtzeitige, und möglichst kostengünstige Auslieferung aller Produkte, unter Berücksichtigung der individuellen Kundenwünsche und -anforderungen. Diese Anforderungen betreffen insbesondere die Aspekte maximale Lieferzeit und Lieferzeitpunkt.

Bezogen auf die **maximale Lieferzeit** gibt es verschiedene Vereinbarungen wie viele (Arbeits-)Tage zwischen dem Auftragseingang und der Auslieferung maximal vergehen dürfen, ohne dass Pönalzahlungen fällig werden. In der Regel kann aber davon ausgegangen werden, dass Betten bzw. Sitzmöbel innerhalb von sechs Wochen ausgeliefert werden müssen. Bei Matratzen gibt es lediglich das Bemühen eine Lieferzeit von zehn Arbeitstagen nicht zu überschreiten, Pönalzahlungen sind hier jedoch nicht vereinbart.

Bei den **Lieferzeitpunkten** gibt es verschiedene Möglichkeiten von Vereinbarungen mit dem Kunden, wobei es der Regelfall ist, dass dem Kunden eine Lieferwoche zugesagt wird, der genaue Liefertag und die Uhrzeit aber von ADA innerhalb der Tourenplanung frei gewählt werden kann. Dabei müssen natürlich die Annahmezeiten, meist „normale" Öffnungszeiten zwischen 7:00 und 17:00 Uhr, berücksichtigt werden. Es gibt jedoch auch Kunden, bei denen Zeitfenster reserviert werden müssen. Der Anteil an Aufträgen bzw. Kunden, bei denen Zeitfenster reserviert werden müssen, wird auf etwa 5% geschätzt. Eine genaue Aussage über diesen Anteil kann derzeit auf Grund der fehlenden Datenaufbereitung nicht getroffen werden. Nach Festlegung des Lieferzeitpunkts muss noch eine Avisierung, das heißt die Bekanntgabe dieses Lieferzeitpunkts, erfolgen. Dies bedeutet, dem Kunden, dem zum Zeitpunkt der Auftragserfassung lediglich die Lieferwoche bekannt gegeben wurde, wird nun der genaue Liefertag mitgeteilt. Ab diesem Zeitpunkt sind Terminverschiebungen natürlich möglichst zu vermeiden.

Die Auslieferung wird überwiegend durch Frächter ausgeführt, wobei es natürlich die Aufgabe der Tourenplanung ist, einen geeigneten, das heißt, einen verlässlichen und kostengünstigen, für jede Tour auszuwählen. Am Standort Anger stehen auch acht eigene Fahrzeuge zur Verfügung. Diese werden jedoch meist für zwischenbetriebliche Transporte innerhalb des Konzerns verwendet. Die Fahrzeuge der verschiedenen Frächter, als auch der eigenen Flotte, unterscheiden sich in Bezug auf die Laderaumkapazität derzeit nicht (oder nur kaum). Es gibt lediglich die zwei Varianten mit oder ohne Hänger.

Wie anfangs erwähnt, stellt die Tourenplanung eine wichtige steuernde Tätigkeit innerhalb des Unternehmens dar. Dies kommt daher, dass die genauen Liefer-, und damit auch Produktionstermine, im Zuge der Tourenzusammenstellung fixiert werden. Dadurch wird die Produktion zum Großteil durch die Tourenplanung getaktet. Auf Grund der kurzen Zeitspanne

zwischen Fixierung des Liefertermins und der tatsächlichen Auslieferung, hat die Produktionsplanung danach nur noch einen kleinen Spielraum für Glättungsmaßnahmen. Deshalb soll innerhalb der Tourenplanung auch auf die Auslastungen der einzelnen Kapazitäten in der Produktion Rücksicht genommen werden. Dies ist jedoch auf Grund der derzeitigen Datenstrukturen kaum realisierbar.

Neben der Planung und Administration der Auslieferungen, zählt es auch zu den Aufgaben der Tourenplanung, die zwischenbetrieblichen Transporte, Abholungen bei Lieferanten, sowie den Rücktransport von Reklamationsware zu bewerkstelligen.

Für den **zwischenbetrieblichen Transport** diverser Güter gibt es für die Werke Anger, Körmend und Nova zwei fixe Touren pro Tag, die Fahrten zwischen diesen Werken und Salonta werden nach Bedarf geplant. Dabei wird meist an diese Fahrt eine Auslieferung von Salonta angehängt.

Da es Lieferanten von Rohstoffen bzw. Zukaufteilen gibt, mit denen eine Selbstabholung vereinbart ist, gibt es auch den Transportauftrag **Abholung beim Lieferanten**. Diese Aufträge werden nicht mit der Tourenzusammenstellung der Auslieferungen mitgeplant, sondern an einer anderen Stelle geplant. Dort wird kontrolliert, ob eine Abholung am Ende einer existierenden Auslieferungstour günstiger als eine einfache Zustellung durch einen Frächter ist.

Da die Transportaufträge für den **Rücktransport von Reklamationsware** meist sehr kurzfristig sind, können diese nicht innerhalb der Tourenzusammenstellung mitgeplant werden. Ist eine Abholung notwendig, wird versucht, diese innerhalb einer bestehenden Tour zu realisieren. Sollte dies nicht möglich sein, so muss ein Zustelldienst beauftragt werden.

Da diese drei Arten von Transportaufträgen jedoch nur einen geringen Anteil ausmachen, und hier kein großes Einsparungspotenzial ersichtlich ist, sind diese explizit kein Bestandteil des Projekts, bzw. dieser Arbeit.

4.2.3 Derzeitige Vorgangsweise

Nachdem nun die wichtigsten Stationen eines Kundenauftrags genannt und die Aufgaben der Tourenplanung erörtert wurden, wird an dieser Stelle die derzeitige Vorgehensweise innerhalb der Tourenplanung kurz geschildert. Dadurch sollen die im nächsten Abschnitt erörterten Ziele des Projekts, die aus der durchgeführten und hier zusammengefassten Ist-Analyse abgeleitet wurden, erklärt werden.

Die Tourenplanung erfolgt derzeit für alle vier Standorte separat, obwohl man sich dem Einsparungspotenzial von Bündelungen mancher Auslieferungen, insbesondere der beiden un-

garischen Standorte, durchaus bewusst ist. Solche gemeinsame Auslieferungen sind jedoch auf Grund der Gegebenheiten derzeit mit erhöhten Aufwand verbunden. Als Beispiele dafür seien hier die intensive Kommunikation zwischen den Beteiligten, sowie die Zusammenführung der notwendigen Papiere (Ladelisten, Etiketten bzw. Warenanhänger und Lieferscheine) genannt. Die Verrechnung der Transportkosten auf die einzelnen Standorte stellt ebenfalls eine gewisse Herausforderung dar.

Da es derzeit keine EDV-Unterstützung für die Tourenplanung gibt, wird alles manuell erledigt. Dies bedeutet, die Kundenaufträge werden in Papierform gesammelt und vom Tourenplaner nach Lieferwoche und Gebiet bzw. Tour sortiert. Dabei werden von ihm Listen geführt, um einen Überblick über die Auslastungen der Laderaumkapazitäten der einzelnen Fahrzeuge zu behalten. Hierfür fehlt jedoch ein geeigneter Zugang zu den im ERP-System hinterlegten Volumenangaben, weshalb der Tourenplaner diese abschätzen muss. Die derzeitig verwendete Einheit für das Volumen eines Produkts ist die sogenannte ADA-Einheit, die dem Volumen einer einfachen Matratze (90x200x18 cm), also ca. 0.33m³, entspricht.

Um die Auslastungen in der Produktion zu berücksichtigen, werden weitere Listen geführt, in denen für jeden Liefertag die Anzahl der Produkte der verschiedenen Produktgruppen eingetragen wird. Da sich die benötigten Zeiten für die Herstellung der verschiedenen Produkte auch innerhalb derselben Produktgruppe stark unterscheiden können, wird dadurch lediglich ein sehr grober Überblick erzielt.

Für die Berücksichtigung der verschiedenen Anforderungen bezüglich des Lieferzeitpunkts, und der damit verbundenen Reservierung diverser Zeitfenster, gibt es ebenfalls keine EDV-Unterstützung – der Tourenplaner muss die unterschiedlichen Anforderungen der Kunden kennen.

Wurde eine Tour festgelegt, müssen die entsprechenden Informationen noch in das ERP-System eingepflegt werden, um die Produktion anzustoßen. Dafür müssen allen Kundenauftragspositionen – ein Kundenauftrag zu einer Sitzgruppe kann zum Beispiel aus den Positionen 3er-Bank, 2er-Bank und Hocker bestehen – der Liefertermin, die Tour-Nummer sowie die Ladeposition (bezüglich der Ladereihenfolge), zugeordnet werden. Diese werden manuell eingetragen.

Anhand dieser kurzen Beschreibung der derzeitigen Vorgehensweise innerhalb der Tourenplanung, kann erkannt werden, dass hierfür ein hoher manueller Aufwand betrieben werden muss, und ein enormes Know-How des Tourenplaners notwendig ist. Aus dieser Tatsache

heraus, die die Kernaussage der durchgeführten Ist-Analyse ist, wurden die im nächsten Abschnitt erörterten Ziele des Projekts abgeleitet.

4.3 Projektvorgaben und Ziele des Konzepts

Wie in Abbildung 11 zu sehen ist, kann als übergeordnetes, zugegebenermaßen sehr allgemein gehaltenes, Ziel die konzernweite Aufwandsreduktion innerhalb der Tourenplanung bei gleichzeitiger Effizienzsteigerung definiert werden. Dieses Ziel kann in vier Unterziele aufgeteilt werden: Verbesserung der Informationsbereitstellung, Erleichterung bei der Tourenzusammenstellung, Zentralisierung der Tourenplanung und Verbesserung der Informationsbereitstellung. Die wichtigsten Eckpunkte dieser vier Teilziele werden in diesem Abschnitt kurz erörtert. Es sei darauf hingewiesen, dass sich diese Unterziele natürlich untereinander beeinflussen, weshalb diese nicht zur Gänze voneinander getrennt werden können.

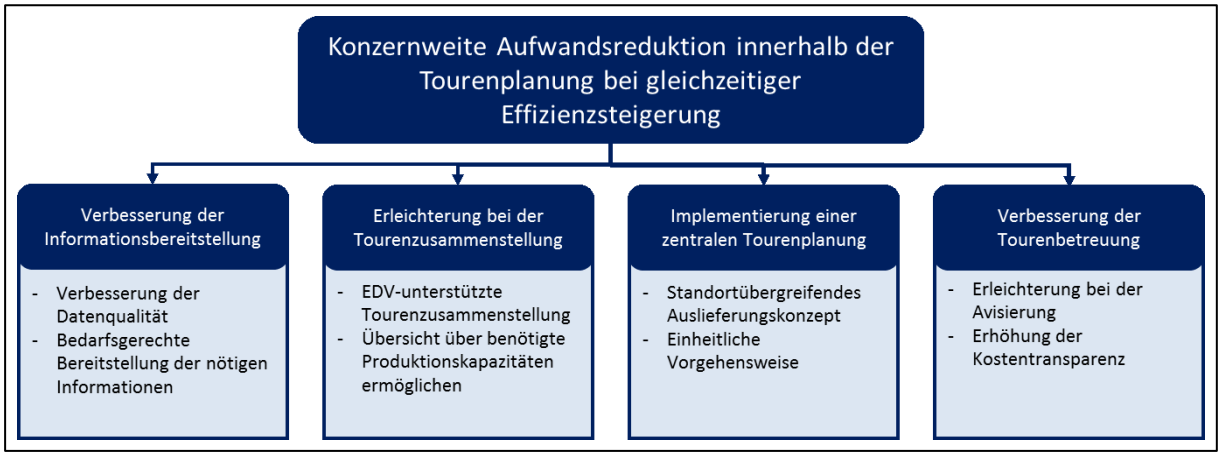


Abbildung 11: Ziele des Projekts

Da eine effiziente Tourenplanung nur möglich ist, wenn alle notwendigen Informationen in geeigneter Form zur Verfügung stehen, stellt die **Verbesserung der Informationsbereitstellung** eine wesentliche Rolle dar. Hierfür müssen zunächst jene Informationen, die derzeit entweder gar nicht, oder nur in einer zur Weiterverarbeitung ungeeigneten Form zur Verfügung stehen (z.B. auf Papier oder in einzelnen Excel-Tabellen), in das ERP-System eingepflegt werden. Neben der Informationsbeschaffung, ist natürlich die bedarfsgerechte Bereitstellung dieser Informationen ein zentrales Thema. Dies bedeutet, die benötigten Informationen sollen dem Tourenplaner in einer übersichtlichen Art und Weise zur Verfügung gestellt werden, wodurch der manuelle Aufwand verringert werden soll.

Auf die **Erleichterung der Tourenzusammenstellung** muss natürlich ein besonderes Augenmerk gelegt werden, da dies die Kernaufgabe der Tourenplanung ist. Hierfür wird ein

Tool benötigt, das erstens den Aufwand für die Zusammenstellung erheblich verringert, und zweitens hilft, eine gute (oder optimale) Tourenzusammenstellung (geringe Transportkosten bei Einhaltung aller Kundenanforderungen und Erfüllung möglichst vieler Kundenwünsche) zu erstellen. Auch die Produktionsauslastungen in den wichtigsten Bereichen sollen mit Hilfe dieses Tools ersichtlich sein. In dieses Tool wird die in dieser Arbeit entwickelte Lösung zur automatischen Tourenzusammenstellung integriert.

Wie sich in der Ist-Analyse gezeigt hat, gibt es derzeit hinsichtlich einer gemeinsamen Auslieferung kaum eine Zusammenarbeit der einzelnen Standorte. Da solch eine Zusammenarbeit jedoch ein erhebliches Einsparungspotenzial besitzt, ist die **Zentralisierung der Tourenplanung** ein wesentlicher Bestandteil des Projekts. Um eine Bündelung der Auslieferungen der verschiedenen Werke realisieren zu können, muss die Administration solcher standortübergreifender Auslieferungstouren erleichtert werden. Hierfür wird neben der Erleichterung durch eine verbesserte Informationsbereitstellung, auch eine Lösung für die im vorigen Abschnitt erwähnten Probleme mit den nötigen Papieren (Ladeliste, Lieferschein, etc.) und der Verrechnung der Transportkosten auf die beteiligten Standorte, benötigt.

Das letzte Unterziel, die **Unterstützung bei der Tourenbetreuung**, bezieht sich auf alle Tätigkeiten, die neben der Tourenzusammenstellung, ebenfalls in den Aufgabenbereich der Tourenplanung fallen. Hier sind insbesondere die Automatisierung der Übernahme der relevanten Informationen aus der Tourenplanung in das ERP-System, die Vereinfachung der Avisierung, sowie eine Erhöhung der Kostentransparenz gefordert.

4.4 Das allgemeine Konzept und die Anbindung der automatischen Tourenplanung

In diesem Abschnitt wird nun das Gesamtkonzept, das auf Basis der eben kurz erörterten Ist-Analyse und den daraus abgeleiteten Zielen, entwickelt wurde, kurz vorgestellt. Dabei wird auf eine ausführliche Beschreibung der umfangreichen organisatorischen Maßnahmen verzichtet. Diese werden lediglich kurz zusammengefasst, um einen Einblick zu geben, wie verstrickt die Tourenplanung in den gesamten Geschäftsprozess ist, und wie viele verschiedene Bereiche innerhalb des Unternehmens von solch einem Projekt betroffen sind.

Das Hauptaugenmerk in diesem Abschnitt soll jedoch auf dem Tool, das für die Tourenzusammenstellung entwickelt wurde, bzw. wird, liegen. Zum Schluss wird noch gezeigt, wie die

automatische Tourenzusammenstellung, die im Zuge dieser Arbeit entwickelt wird, in dieses Tool, bzw. in den gesamten Tourenplanungsprozess, integriert werden soll.

4.4.1 Organisatorische Maßnahmen

Eine der wesentlichsten Maßnahmen innerhalb dieses Projekts betrifft die Informationsbeschaffung, sowie die Datenaufbereitung und -wartung. Wie in der Beschreibung der derzeitigen Vorgehensweise gezeigt, betrifft dies verschiedenste Bereiche. Die wichtigsten Informationen, die in die Stammdaten des ERP-Systems neu eingepflegt werden müssen, sind die Warenannahmezeiten der einzelnen Kunden (bzw. ob Zeitfenster reserviert werden müssen) sowie deren Kontaktdaten (um die Avisierung automatisieren zu können). Diese Informationen werden mit Hilfe der Frächter in Erfahrung gebracht bzw. gesammelt. Da im ERP-System noch nicht allen Produkten Volumen zugeordnet sind, diese aber natürlich für eine Berechnung der Laderaumauslastung benötigt werden, wird auch eine diesbezügliche laufende Kontrolle implementiert.

Eine weitere wichtige Aufgabe im Bereich der Datenaufbereitung ist die Harmonisierung der Lieferadressen aller vier Standorte. Diese werden an allen vier Standorten separat verwaltet, obwohl sich die Kunden der verschiedenen Werke kaum unterscheiden. Durch diese separate Verwaltung kommt es, dass dieselben Adressen mehrfach (teilweise auch innerhalb eines Werkes), in unterschiedlichen Schreibweisen, vorhanden sind. Da die Kundenaufträge auf die Lieferadressen zusammengefasst werden können sollen, müssen natürlich gleiche Lieferadressen als solche erkannt werden. Die Anzahl an vorhandenen Lieferadressen liegt bei jedem Standort bei etwa 3.000, weshalb eine manuelle Synchronisation über eine einheitliche, eindeutige Schreibweise nicht (oder nur sehr schwer) realisierbar ist. Deshalb soll diese Harmonisierung über die Geokoordinaten erfolgen. Dafür wurde eine kleine Hilfs-Applikation erstellt, die die Lieferadressen aller zukünftigen Aufträge, deren Koordinaten noch nicht bestätigt und in die Stammdaten geschrieben wurden, überprüft. Diese werden geocodiert und müssen einmalig kontrolliert und bestätigt werden. Dies hat neben der Eindeutigkeit noch weitere Vorteile, auf die im nächsten Abschnitt näher eingegangen wird.

Für die Anbindung des Tourenplanungs-Tools an das vorhandene ERP-System, wird von dessen Hersteller eine Schnittstelle entwickelt, die zu allen Kundenaufträgen eine Referenz enthält. In diese können alle, innerhalb der Planung relevanten Informationen, gespeichert werden. Um die Produktionsauslastungen innerhalb der Tourenplanung mit berücksichtigen zu können, werden von jedem Werk die drei wichtigsten Arbeitsbereiche, inklusive deren Kapazität, ausgewählt. Diese werden dem Tourenplaner zur Verfügung gestellt.

Als eine der wichtigsten Maßnahmen, die nicht der Informationsbeschaffung und Datenverarbeitung zugeordnet werden können, sei die Anpassung diverser Listen und Begleitscheine zu nennen. Dies ist insbesondere auf Grund der standortübergreifenden Auslieferungstouren nötig.

4.4.2 Das Tourenplanungs-Tool

An dieser Stelle wird nun kurz das „Herzstück" des Projekts, das entwickelte Tourenplanungs-Tool, vorgestellt. Da eine etwas ausführliche Beschreibung dieses Tools ein eigenes Kapitel füllen würde, wird hier lediglich auf die wichtigsten Funktionalitäten und die Anbindung an das vorhandene ERP-System und damit an die gesamte Auftragsabwicklung, eingegangen. Dadurch werden einerseits die wesentlichen Verbesserungen, die durch Einsatz dieses Tools erzielt werden, ersichtlich, und andererseits wird gezeigt, wie dieses Tool in den Geschäftsprozess integriert wird.

Das Tool wurde in der Programmiersprache JAVA geschrieben, wobei für die Verarbeitung der Geo-Informationen (Geocodierung, Kartendarstellung, Entfernungen, etc.) ein Produkt der Firma PTV AG, das für die Integration in eigene Entwicklungen vorgesehen ist, zugekauft wurde. Die Applikation wurde als Client-Anwendung konzipiert, wobei das eben erwähnte Produkt als Web-Dienst auf einem, für alle vier Standorte zentralen, Server zur Verfügung steht. Auf diesem Server sind auch, neben diversen Konfigurationsdateien, zwei Matrizen, die die Distanzen bzw. die Fahrzeiten zwischen den bekannten (überprüften) Lieferadressen bzw. Geokoordinaten enthalten. Diese und noch weitere Daten werden durch eine eigene Applikation jede Nacht mit den Daten des ERP-Systems abgeglichen, und entsprechend aktualisiert. Diese nächtliche Generierung bzw. Aktualisierung der Distanzmatrizen, die durch die im vorigen Abschnitt erwähnte Kontrolle der Geokoordinaten ermöglicht wird, bringt zwei entscheidende Vorteile mit sich: Erstens wird dadurch die Performance bei der manuellen Tourenplanung erheblich verbessert, da der Web-Dienst nicht jedes Mal die Distanzen innerhalb einer Tour neu errechnen muss. Zweitens steht dadurch die Möglichkeit offen, eine „interaktive" Optimierung der Tourenzusammenstellung in das Programm zu integrieren. Würde man diese Matrizen nicht zentral verwalten, müssten sie für die Optimierung zum Zeitpunkt der Ausführung generiert werden, dies dauert jedoch für 500 Adressen in etwa 45 Minuten, wobei der Web-Dienst natürlich entsprechend beansprucht wird.

Die Kommunikation mit dem ERP-System, und damit der Zugang zu allen relevanten (Auftrags-)Daten, erfolgt über die bereits erwähnte Schnittstelle. Beim Start der Tourenplanung werden für die ausgewählte Kalenderwoche alle Aufträge aus der Schnittstelle (und die da-

zugehörigen, relevanten Informationen aus den diversen Dateien des ERP-Systems) geladen. Diese werden anschließend lokal verarbeitet bzw. entsprechend aufbereitet, wobei es auch möglich ist, mehrere Kalenderwochen parallel zu bearbeiten.

Wie in Abbildung 12 (die Screenshots des entwickelten Tools zeigt) ersichtlich ist, ist die Applikation auf die Verwendung mit zwei Bildschirmen ausgelegt. Die Bedienung ist natürlich auch mit einem Bildschirm möglich, jedoch unkomfortabler, da zwei Fenster verwendet werden. Eines davon wird als Informationsfenster verwendet, das abhängig von den ausgeführten Funktionen des anderen, die Anzeige verändert.

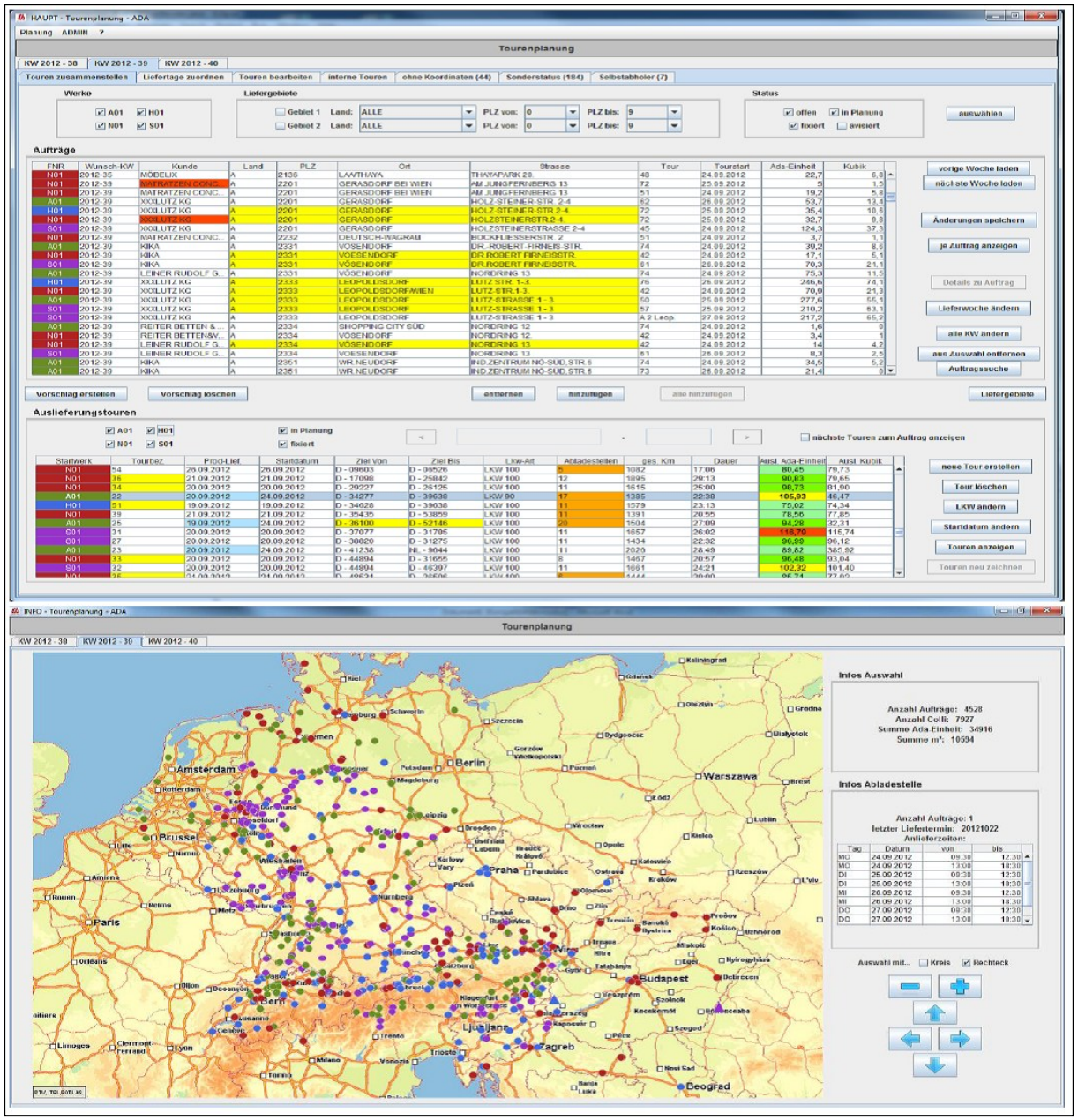


Abbildung 12: Screenshot „Tourenzusammenstellung"

In obiger Abbildung sind die zwei Fenster, wie sie in der Startansicht gezeigt werden, abgebildet. Dabei sind im Funktionsfenster alle Aufträge und bereits geplanten Touren in Tabellen

gelistet. In dieser Ansicht können zum Beispiel Touren zusammengestellt bzw. geändert, oder diverse Informationen abgefragt, werden. Dabei ist eine Selektion der angezeigten Aufträge nach verschiedenen Kriterien (z.B. Standort, Lieferadresse oder Auftragsstatus), oder durch Markierung eines Gebietes auf der Landkarte, möglich. Das Informationsfenster enthält eine Landkarte, auf der die Aufträge, sowie die bereits geplanten Touren als Polygone angezeigt werden können. Zusätzlich sind diverse Informationen zu der derzeitigen Auftragsauswahl und dem in der Tabelle markierten Auftrag, ersichtlich.

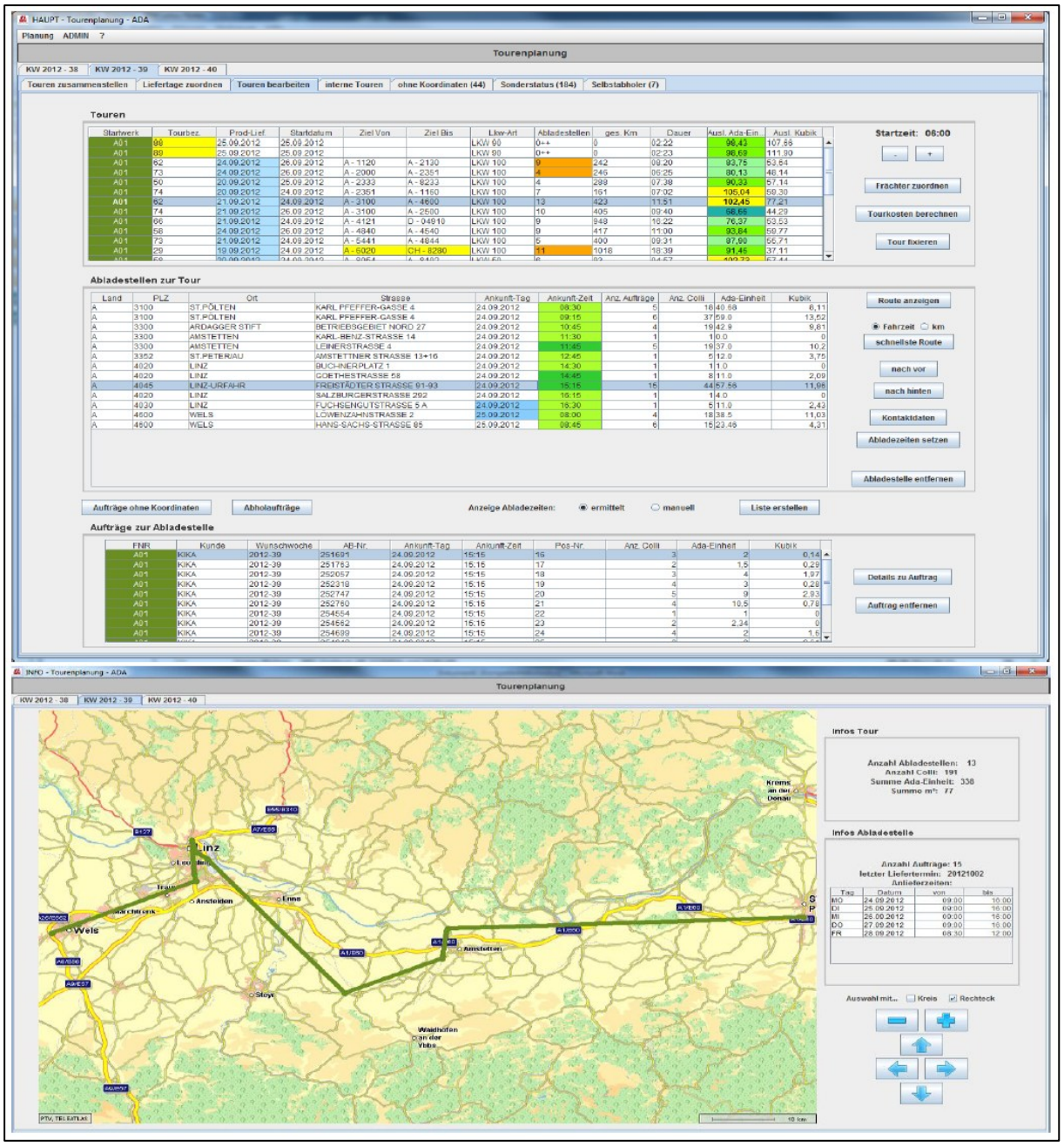


Abbildung 13: Screenshot „Touren bearbeiten"

In einer weiteren Ansicht, die in Abbildung 13 gezeigt wird, können die einzelnen Touren bearbeitet werden. In dieser werden zur ausgewählten Tour die dazugehörigen Abladestel-

len, und zu diesen, die dazugehörigen Kundenaufträge, angezeigt. Dabei werden die errechneten Ankunftszeiten der einzelnen Abladestellen mit den im System hinterlegten Öffnungszeiten verglichen, und entsprechend in der Tabelle eingefärbt. Dadurch soll auf mögliche Zeitfensterverletzungen aufmerksam gemacht werden. Für die Reihenfolge der Abladestellen der gewählten Tour, die im Informationsfenster als Polygon angezeigt wird, kann per Knopfdruck ein Vorschlag erstellt werden. Dieser wird nach dem Lin-Kernighan-Verfahren ermittelt (siehe Abschnitt 5.2.2). Natürlich gibt es auch die Möglichkeit, die Reihenfolge manuell festzulegen.

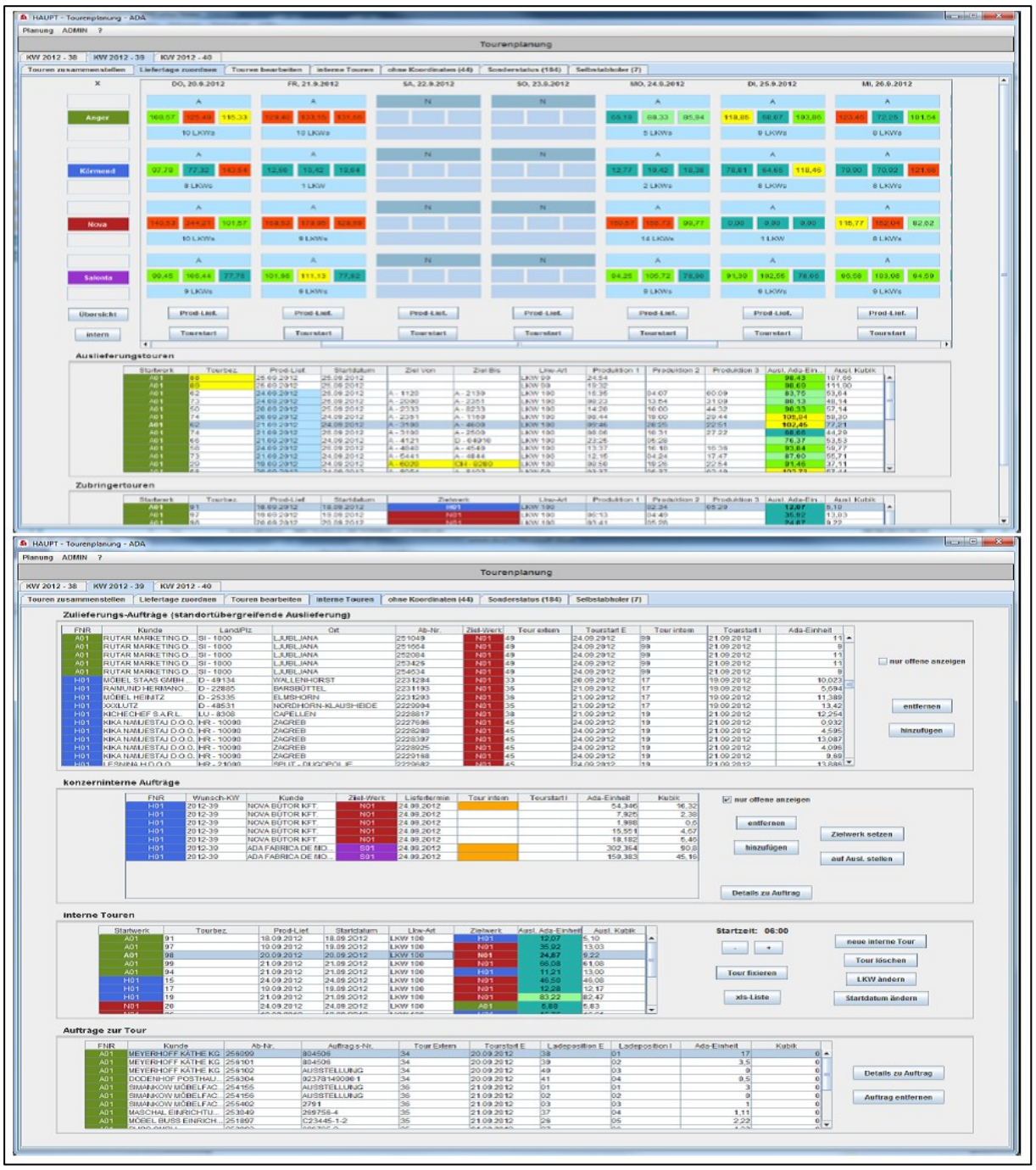


Abbildung 14: Screenshot "Produktion" und "interne Touren"

Zwei weitere wichtige Bereiche innerhalb dieser Applikation sind in Abbildung 14 auf der vorigen Seite gezeigt. Das obere Bild zeigt die Übersicht der Auslastungen der wichtigsten Produktionsbereiche in den verschiedenen Werke. Hier können die Touren den einzelnen Tagen zugeordnet werden, wobei darauf aufmerksam gemacht wird, wenn bei einer Terminverschiebung eine Restriktion verletzt wird. Im unteren Bild ist die Anzeige für die zwischenbetrieblichen Transporte abgebildet. Wird bei der allgemeinen Tourenzusammenstellung (Abbildung 12) ein Auftrag eines Werkes, einer Tour eines anderen Standorts zugeordnet, so muss dieser natürlich zunächst mit einer zwischenbetrieblichen Tour zu diesem Werk transportiert werden. Solche Aufträge sind in dieser Ansicht zusammengefasst, und hier kann die Zuteilung zu entsprechenden konzerninternen Touren erfolgen.

Anhand dieser kurzen Beschreibung des Tools kann bereits festgestellt werden, dass damit die Tourenplanung innerhalb des Unternehmens natürlich erheblich erleichtert wird. Damit werden eigentlich bereits alle Ziele, die im vorigen Abschnitt erwähnt wurden, erreicht: Die Informationsbereitstellung wird entscheidend verbessert, die Tourenzusammenstellung erleichtert, eine zentrale Planung ermöglicht und die Tourenbetreuung verbessert. Letzteres wird insbesondere dadurch erreicht, dass aus der eben vorgestellten Applikation die Übertragung der relevanten Informationen an das ERP-System automatisch erfolgt. Weiters werden zu jeder Tour Frächter und Preis, sowie kalkulierte Kosten (inklusive anfallender Mautgebühren), festgehalten, um dadurch eine erhöhte Kostentransparenz zu erhalten.

Um jedoch auch die Effizienz der Tourenzusammenstellung zu erhöhen, wird in dieses Tool die Möglichkeit integriert, diese automatisch durchzuführen. Die Entwicklung dieser Funktion wurde als Hauptthema dieser Arbeit gewählt. Natürlich kann ein solche Zusammenstellung vom Benutzer weiter verändert werden. Bevor im nächsten Kapitel diese Entwicklung der automatischen Tourenzusammenstellung beschrieben wird, wird kurz erörtert, wie diese in dieses Tool, und damit in die gesamte Tourenplanung, integriert werden soll.

4.4.3 Integration der automatischen Tourenzusammenstellung

Innerhalb des eben kurz vorgestellten Tools soll es dem Tourenplaner möglich sein, für den gewählten Zeitraum (also für die entsprechende Kalenderwoche) eine optimierte Tourenzusammenstellung zur weiteren Bearbeitung abzurufen. Für dieses Abrufen eines Vorschlags stehen zwei grundlegende Vorgangsweisen zur Auswahl. Eine Möglichkeit ist es, die Optimierung für die kommenden Wochen immer nachts durchzuführen, und die Ergebnisse abzuspeichern. Damit wäre natürlich die Anzeige des Vorschlags ohne weitere Rechenzeiten realisierbar, und auf die Rechenzeit des Verfahrens muss nicht wirklich geachtet werden (sofern

sich natürlich noch ein Durchlauf in der Nacht ausgeht). Der Nachteil ist natürlich, dass Informationen bzw. Aufträge des aktuellen Tages noch nicht berücksichtigt sind. Die andere Möglichkeit wäre ein manuelles Anstoßen der Optimierung direkt in der Tourenplanungs-Applikation. Diese Möglichkeit ist aus vielen Gründen zu bevorzugen, denn erstens werden somit aktuelle Informationen berücksichtigt, und zweitens kann der Tourenplaner selbst entscheiden, wann welche Aufträge in einem Optimierungslauf verplant werden sollen. Somit ist es dem Tourenplaner möglich, verschiedene Szenarien auszuprobieren, die eventuell von einem Optimierungsalgorithmus nicht erkannt, bzw. für diesen nicht erlaubt, sind. So kann der Tourenplaner zum Beispiel entscheiden, die Liefertermine (bzw. die Lieferwoche) mancher Aufträge zu verschieben um eine bessere Tourenzusammenstellung zu ermöglichen – eine Entscheidung, die von einem automatischen Verfahren nicht getroffen werden kann (und vor allem soll). Um diese, in dieser Arbeit angestrebte, Variante realisieren zu können, muss jedoch die Rechenzeit des Verfahrens sehr gering sein, denn ansonsten kann ein komfortables Interagieren mit dem Programm nicht ermöglicht werden.

Unabhängig davon, wie und wann ein Vorschlag erstellt wird, soll der ermittelte Vorschlag zwar von dem Tourenplaner völlig frei bearbeitet und verändert werden können, bezüglich der Tourenzusammenstellung soll aber eine Änderung des Vorschlags nicht mehr nötig sein. Um dieses Ziel zu erreichen, müssen fünf wichtige Fragen geklärt werden: Wie werden Kunden, bei denen Zeitfenster reserviert werden müssen, entsprechend berücksichtigt? Wie erfolgt die Einplanung von kurzfristigen Aufträgen? Wie wird die standortübergreifende Bündelung von Auslieferungen einbezogen? Wie werden bereits fixierte Touren (oder auch geplante Touren) in den Vorschlag integriert und wie werden die Touren auf die einzelnen Tage innerhalb der Kalenderwoche aufgeteilt?

Im Zuge der Implementierung des eben beschriebenen Tools, und der damit verbundenen intensiven Beobachtung der Tourenplanung, hat sich herausgestellt, dass die **Berücksichtigung von Zeitfenstern** gar kein großes Problem darstellt. Da die Tourenplanung immer mindestens zwei Wochen im Voraus erfolgt, sind nach Abschluss der Planung noch genügend „passende" Zeitfenster vorhanden. Dies bedeutet, die Tourenplanung erstellt zunächst die Touren (und Routen) ohne Berücksichtigung von Zeitfenstern und erst danach werden die aus den geschätzten Ankunftszeiten abgeleiteten Zeitfenster beim Kunden reserviert. Damit macht es innerhalb der Planung keinen Unterschied, ob bei einem Kunden Zeitfenster reserviert werden müssen oder nicht.

Derzeit wird für kurzfristige Aufträge, also in der Regel Matratzen, teilweise Laderaum freigelassen. Deshalb soll auch bei der automatischen Zusammenstellung eine **Einplanung kurz-**

fristiger Aufträge erfolgen. Dafür soll vom Tourenplaner je Liefergebiet ein Prozentsatz angegeben werden können, der aussagen soll, mit welcher Menge an Aufträgen noch zu rechnen ist. Als Hilfestellung wird es auch auf Basis der Durchlaufzeiten der vergangenen Wochen einen Vorschlag für diesen Wert geben. Die kurzfristigen Aufträge werden somit durch eine prozentuelle Erhöhung des Volumens der bereits existierenden Aufträge berücksichtigt.

Da das Ausmaß der Einsparungen durch eine **standortübergreifende Bündelung von Auslieferungen** derzeit nur schwer kalkulierbar ist, wird die Entscheidung, wann welche Aufträge standortübergreifend ausgeliefert werden, nicht innerhalb des Optimierungsprozesses, getroffen. Stattdessen wird vom Tourenplaner festgelegt, in welchen Liefergebieten von welchen Standorten eine gemeinsame Auslieferung sinnvoll ist. Die betroffenen Aufträge werden dem entsprechenden Standort, von dem aus die Belieferung erfolgt, zugeordnet. Zum Schluss der Optimierung werden dann für diese Aufträge die konzerninternen Zuliefertouren erstellt. Durch diese Aufteilung zu Beginn der Optimierung entstehen vier unabhängige Ein-Depot-Probleme, die parallel gelöst werden können. Erst wenn die Auswirkungen gemeinsamer Auslieferungen genauer bekannt sind, ist eine mögliche Integration dieses Entscheidungsprozesses in die Optimierung denkbar.

Es besteht natürlich die Möglichkeit, dass zum Zeitpunkt des Abrufs eines Vorschlags bereits eine fixierte Tour innerhalb der gewählten Kalenderwoche existiert. Deshalb muss auch eine **Einbeziehung von bereits geplanten bzw. fixierten Touren** in den Vorschlag möglich sein. Dies soll einfach dadurch geschehen, dass weder diese Touren, noch die diesen zugeordneten Aufträge, berücksichtigt werden. Eine solche Vorgehensweise vereinfacht nicht nur die Implementierung eines Optimierungs-Verfahrens, sondern ermöglicht auch das oben erwähnte Testen von verschiedenen Szenarien.

Der letzte wichtige Punkt den es zu berücksichtigen gilt, ist die **Zuteilung der Touren auf die einzelnen Tage**. Da der überwiegende Teil der Touren innerhalb der Woche frei verschoben werden kann, reicht es aus, die generierten Touren erst am Ende auf die Tage aufzuteilen. Dabei muss berücksichtigt werden, dass alle Aufträge bis zum Ende der Woche ausgeliefert sind, also mehrtägige Touren nicht erst am Ende der Woche starten. Als Ziel der Aufteilung fungiert natürlich die möglichst ausgeglichene Produktionsauslastung.

5 Die automatische Tourenzusammenstellung

In den ersten beiden Kapitel wurden das Vehicle Routing Problem sowie die wichtigsten Ansätze zur Lösung dieses Problems erörtert. Darauf aufbauend wird nun an dieser Stelle eine Lösung für die im vorigen Kapitel beschriebene Problemstellung erarbeitet. Zunächst wird kurz darauf eingegangen, wie bei der Entwicklung der Lösung vorgegangen wird, bzw. nach welchen Kriterien die verschiedenen Lösungsansätze bewertet, verglichen und ausgewählt werden. Anschließend werden Lösungen für das Routing-Problem analysiert und danach werden Konstruktionsheuristiken sowie einfache lokale Suchverfahren für die Lösung des Gesamtproblems untersucht. Am Schluss werden noch Verbesserungsverfahren gezeigt, mit deren Hilfe noch bessere Lösungen gefunden werden sollen.

5.1 Vorgehensweise bei der Entwicklung der Lösung

Die Möglichkeit zur automatischen Tourenzusammenstellung soll natürlich in das im vorigen Kapitel gezeigte Tourenplanungsprogramm integriert werden. Da dieses aber bereits im Echtbetrieb läuft, und sich zum Testen und Auswerten von Optimierungsalgorithmen ohnehin nicht wirklich eignet, wurde eine eigene Hilfsapplikation für die Entwicklung programmiert. Erst die endgültige Lösung wird in das tatsächlich verwendete Programm eingearbeitet. Diese Hilfsapplikation zur Entwicklung der Lösung wird im folgenden Abschnitt vorgestellt. Davor sollen aber die generellen Überlegungen zum Aufbau der Lösung kurz erörtert werden.

Wie bereits im vorigen Kapitel kurz angedeutet, soll die Aufteilung der Aufträge auf die verschiedenen Werke, das heißt die Auswahl der Liefergebiete, die von mehreren Werken gemeinsam beliefert werden, vor dem eigentlichen Optimierungsprozess erfolgen. Damit entstehen vier unabhängige, bezüglich dem Aufbau idente, Problemstellungen. Diese Aufteilung des Problems hat neben der Vereinfachung vor allem auch den Vorteil, dass die vier Standorte weiterhin separat planen können. Ebenfalls bereits angedeutet wurde die Tatsache, dass die Zeitrestriktionen innerhalb der Planung nicht als harte Zeitfenster betrachtet werden müssen (da die Reservierung der Zeitfenster ohnehin erst nach der Planung stattfindet). Die Einhaltung der Öffnungs- sowie Lenk- und Ruhezeiten wird ebenfalls nicht explizit geprüft. Diese Vereinfachung ist deshalb zulässig, da man sich ohnehin Frächtern bedient und somit keine Kenntnis über die bereits absolvierte Fahrleistung des Fahrers vor dem Start der Tour hat. Auch der Zentraldisponent des Unternehmens konnte bestätigen, dass es keine Preis-

sprünge bei Überschreitung der maximalen Tageslenkzeiten (die sich mehr oder weniger mit den Standard-Öffnungszeiten decken) gibt.

5.1.1 Die Hilfs-Applikation für die Entwicklung der Algorithmen

Die Applikation zur Entwicklung und Analyse der verschiedenen Lösungsansätze wurde, wie auch das eigentliche Tourenplanungsprogramm, in der Programmiersprache JAVA geschrieben. Diese Applikation ist so aufgebaut, dass mit Hilfe der Distanz- bzw. Fahrzeit-Matrix der realen Kunden sowohl TSP- als auch VRP-Instanzen erzeugt werden können. Als Parameter werden Knotenanzahl, Bandbreite der Einzelbedarfe sowie die zur Verfügung stehenden Fahrzeuge eingegeben. Dadurch können die Algorithmen auf die verschiedensten Ausprägungen von Probleminstanzen getestet werden. Die Algorithmen werden alle in einer Form programmiert, bei der die wesentlichen Einstellungen des Algorithmus beim Aufruf als Parameter übergeben werden können. Bei jedem Testlauf wird ein Protokoll geführt, das die relevanten Informationen festhält (Parametereinstellungen, beste Lösung, Rechenzeit, Verlauf der Suche, etc.). Diese Protokolle werden gemeinsam mit der Testinstanz gespeichert, und können, wie in Abbildung 15 ersichtlich, angezeigt und verglichen werden. Neben den wichtigsten Daten zu den einzelnen Testläufen, wird das Ergebnis auch auf einer Karte angezeigt. Dies ist insbesondere deshalb wichtig, da nur über diese „Sichtkontrolle" (in Verbindung mit der angezeigten Laderaumauslastung) die Qualität der verwendeten Kostenfunktion abgeschätzt werden kann (siehe nächsten Abschnitt).

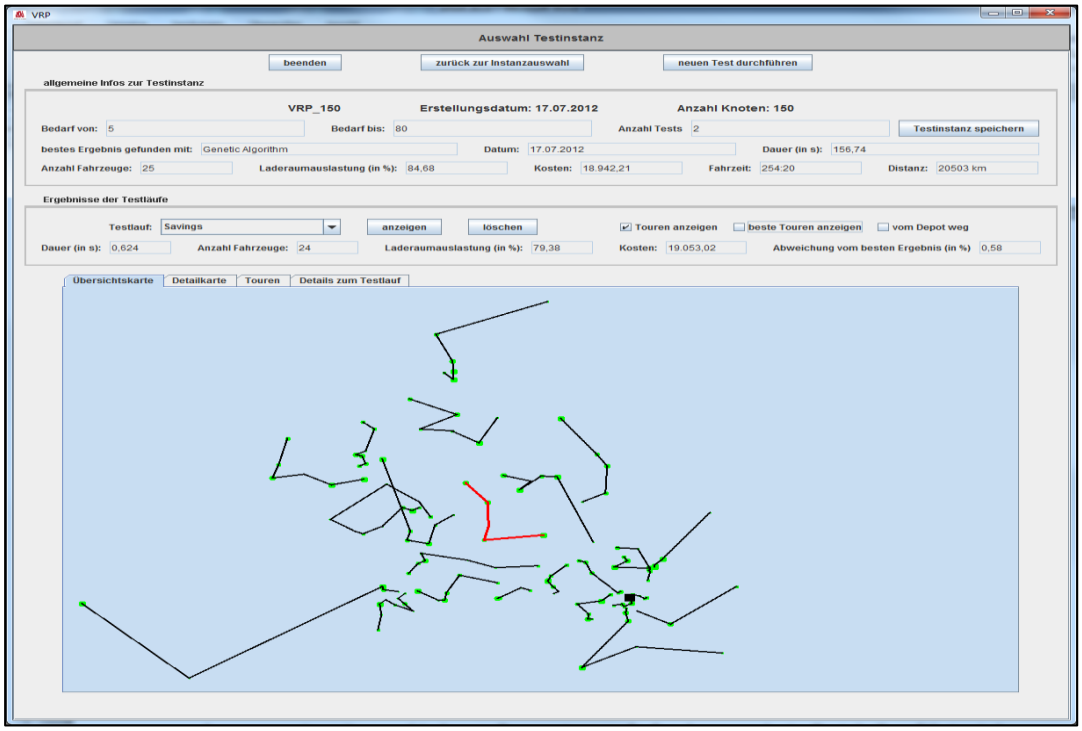


Abbildung 15: Screenshot Testapplikation - Tourenanzeige

Da nicht nur das Endergebnis sondern auch der Verlauf der Suche für die Qualität eines Testlaufs von Bedeutung ist, kann man sich diesen auch in Form von Diagrammen anzeigen lassen (siehe Abbildung 16).

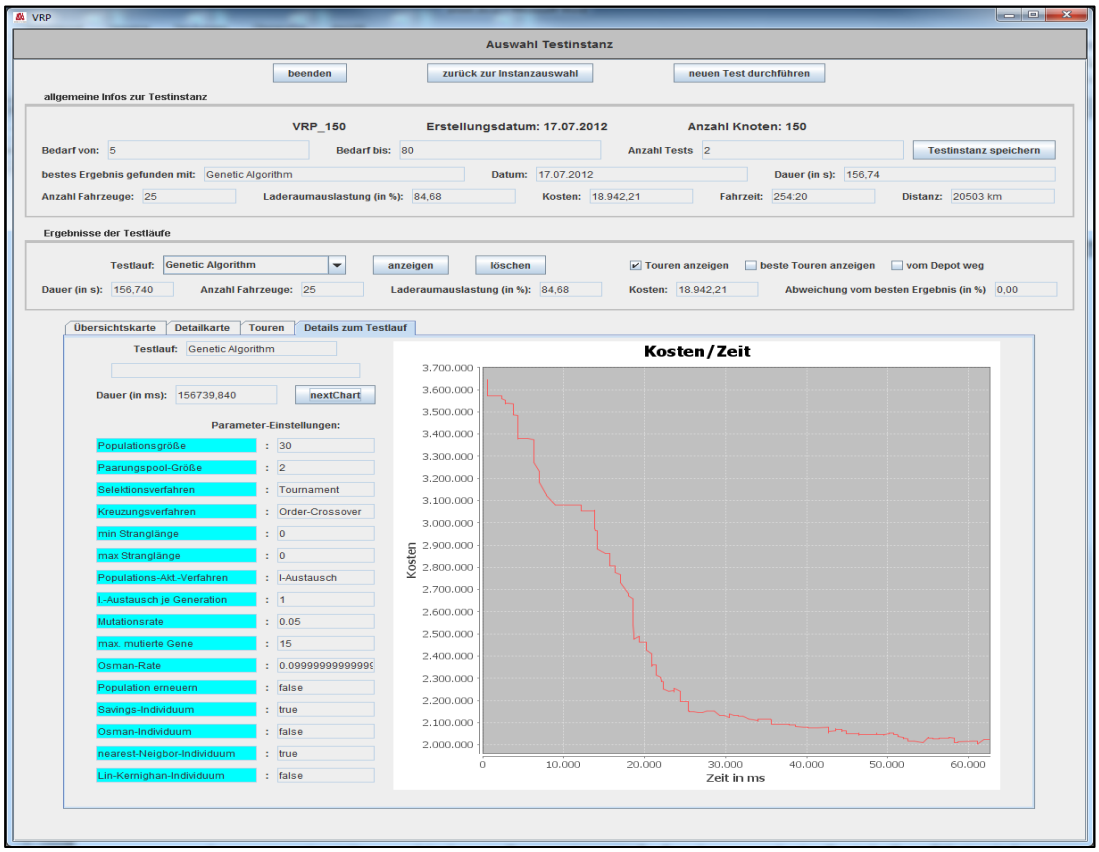


Abbildung 16: Screenshot Testapplikation - Anzeige Suchverlauf

5.1.2 Bewertungs- und Vergleichskriterien – die Entwicklung der Zielfunktion

Um die verschiedenen Lösungsansätze objektiv bewerten und vergleichen zu können, muss erstens bekannt sein, wie die hier entwickelte Lösung in den Tourenplanungsprozess integriert werden soll, und zweitens wie sich die Kosten eines gesamten Tourenplans zusammensetzen. Es wurden im vorigen Kapitel grundsätzlich zwei Möglichkeiten aufgezeigt, wie ein Optimierungsprozess in die Tourenplanung integriert werden kann: Ein ausführlicher Optimierungslauf in der Nacht (bei dem somit die Laufzeit eine sehr untergeordnete Rolle spielt) oder aber ein manuelles Anstoßen aus dem Tourenplanungsprogramm heraus. Hier ist, wie bereits erwähnt wurde, sicherlich die zweite Variante zu bevorzugen, da damit der Tourenplaner mit dem Optimierungstool interagieren kann. Solch eine Interaktion hätte vor allem den Vorteil, dass verschiedene Szenarien getestet und verglichen werden können. Dies bedeutet, der Bediener des Programms macht mehrere Testläufe mit verschiedenen Zusam-

mensetzungen der Aufträge, wobei diese unterschiedlichen Zusammensetzungen z.B. durch Veränderung der Lieferwoche mancher Aufträge, durch Verschiebung der standortübergreifend belieferten Gebiete, oder aber durch Ausschluss bestimmter Aufträge (z.B. Speditionsversand) generiert werden. Weiters wäre es durchaus denkbar, dem Bediener die Möglichkeit zu geben, die Gewichtungen der einzelnen Faktoren der Kostenfunktion zu verändern, bzw. ihm verschiedene Varianten der Optimierung zur Verfügung zu stellen.

Durch die eindeutige Präferenz der Variante des manuellen Anstoßes des Optimierungstools, wird als erstes wichtige Bewertungskriterium die Laufzeit der verschiedenen Algorithmen herangezogen. Diese ist natürlich leicht mess- und vergleichbar, da alle Testläufe mit demselben Notebook (2,3 GHz Dual-Core Prozessor, 6 GB RAM, 64 Bit-Betriebssystem – Windows 7) durchgeführt werden. Das zweite wichtige Bewertungskriterium ist natürlich die Lösungsqualität, die durch den (zu minimierenden) Zielfunktionswert repräsentiert wird.

Die Erstellung einer geeigneten Zielfunktion stellt wahrscheinlich den Schlüssel zu einem in der Praxis tatsächlich verwendbaren Optimierungstool dar. So konnte zum Beispiel der Zentraldisponent aus Erfahrung berichten, dass in einem anderen möbelproduzierenden Unternehmen das vorhandene Optimierungstool deren Tourenplanungsprogramms (eines namhaften Herstellers) nicht verwendet wird, da aus deren Sicht die Lösungen nicht praktikabel sind. Dies wird sicherlich nicht auf schlechte Optimierungsalgorithmen zurückzuführen sein, sondern vielmehr auf eine unzureichende Berücksichtigung diverser Kriterien innerhalb der verwendeten Zielfunktion. Deshalb soll der Entwicklung der Zielfunktion an dieser Stelle besondere Aufmerksamkeit geschenkt werden:

Grundsätzlich gilt, dass möglichst wenige Fahrzeuge eine möglichst geringe Gesamtdistanz zurücklegen sollen. Doch die alleinige Berücksichtigung dieser beiden Faktoren (unabhängig von der Gewichtung bzw. ob die Fahrzeit, die Distanz oder beides herangezogen wird) führt zu Lösungen, die in der Praxis in dieser Form sicherlich nicht umgesetzt werden. Denn wie in Abbildung 17 a) auf der nächsten Seite zu sehen ist, werden dadurch Touren vorgeschlagen, die sich über große Entfernungen ziehen. In diesem Fall würde zum Beispiel die rotmarkierte Tour über Wien nach Berlin und weiter in den Hamburger Raum führen. Solche Touren werden in der Praxis jedoch nicht gefahren, denn dadurch verliert der Frächter erstens an Flexibilität (da er die Abladereihenfolge nicht mehr kurzfristig ändern kann) und zweitens wird die Planung der Abladezeiten wesentlich schwieriger. Aus diesem Grund wird in die Kostenfunktion ein Faktor integriert, durch den „kompakte" Touren bevorzugt werden. Hierfür wurden verschiedene Varianten mit demselben Algorithmus getestet. Abbildung 17 b) auf der nächsten Seite zeigt zum Beispiel eine Variante, bei der die distanzabhängigen Kos-

ten für die Teilstrecke vom Depot zum ersten Kunden geringer sind als für die restliche Strecken – dadurch werden natürlich langgezogene Touren als ungünstiger angesehen. Der Teil der Kostenfunktion, der für die Bevorzugung kompakter Touren verantwortlich ist, wird im weiteren Verlauf dieser Arbeit als Clusterfaktor bezeichnet.

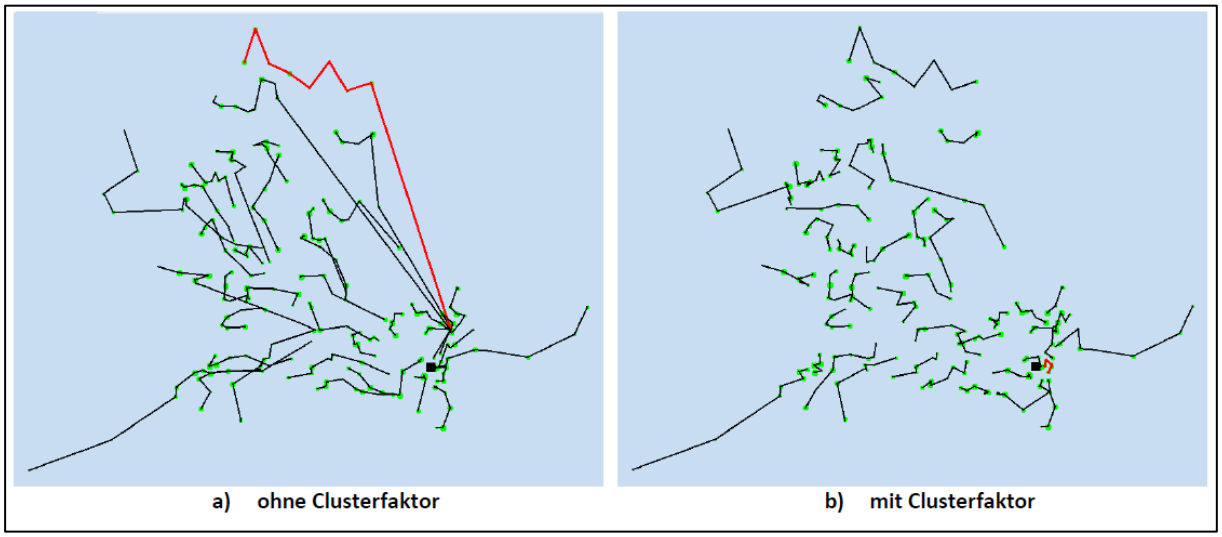


Abbildung 17: Ergebnis mit und ohne Clusterfaktor

Wie bereits kurz angedeutet, kann als Gesamtdistanz der Touren sowohl die Summe der Fahrzeiten, die Summe der Distanzen oder aber auch eine Kombination dieser beiden Werte herangezogen werden. Zusätzlich gibt es die Möglichkeit, auch die Dauer der Entladung zu berücksichtigen. Tests haben jedoch gezeigt, dass sich durch Veränderung der Distanzkostenermittlung lediglich die absoluten Kostenfunktionswerte verschiedener Lösungen ändern, nicht aber deren relative Größe zueinander. Dies ist aber nicht weiter verblüffend, denn die Fahrzeit ist zur Distanz direkt proportional und die Gesamtdauer aller Entladungen ist konstant. Aus diesem Grund wird innerhalb der verwendeten Kostenfunktion lediglich die Distanz herangezogen. Somit besteht die Kostenfunktion aus einem Fixanteil je LKW, aus den distanzabhängigen Kosten sowie dem Clusterfaktor. Um die verschiedenen Fahrzeugarten zu berücksichtigen, können sowohl die Fixkosten als auch die distanzabhängigen Kosten für jeden Fahrzeugtyp separat definiert werden (da z.B. ein LKW ohne Hänger eine geringere Maut entrichten muss als ein LKW mit Hänger).

Je nach Gewichtung der drei Faktoren wird die Lösung in eine andere Richtung gedrängt. So führt zum Beispiel eine Erhöhung der Fixkosten je Fahrzeug natürlich zu einer Präferenz von Lösungen mit hoher Laderaumauslastung (zum Nachteil der Gesamtdistanz und der Clusterung). Das heißt, es gibt drei konkurrierende Ziele, wobei versucht werden muss, einen ausgewogenen Zustand zu finden. Da dieser aber nicht eindeutig erkennbar ist, und auch von der subjektiven Betrachtung des Tourenplaners abhängt, sollte es, wie bereits erwähnt, die Möglichkeit geben, für jeden Testlauf die Gewichtungen dieser Faktoren zu verändern.

5.2 Lösungen für das Routing

Da für jeden Lösungsansatz des Gesamtproblems auch eine Lösung für das Routen der einzelnen Touren benötigt wird, sollen an dieser Stelle verschiedene TSP-Lösungen getestet und verglichen werden. Wie bereits bei der Beschreibung der bekannten Ansätze zur Lösung solcher Probleme erwähnt wurde, hat man es innerhalb einer Tour eines VRPs selten mit einer großen Anzahl an Abladestellen, also Knoten, zu tun. In diesem speziellen Fall geht die Anzahl an Abladestellen nur sehr selten über 30, was einerseits auf die voluminösen Produkte und andererseits auf die größeren Bestellmengen zurückzuführen ist (Belieferung der Möbelhäuser anstatt der Endkunden!). Auf Grund dessen, dass bei den meisten Verfahren zur Lösung des VRPs sehr viele Zusammenstellungen von Abladestellen getestet werden müssen, soll insbesondere untersucht werden, ab welcher Anzahl an Knoten eine Optimierung der Reihenfolge notwendig bzw. sinnvoll ist, und mit welchem Verfahren dies möglichst zuverlässig und rasch erfolgen kann. Dafür werden die ausgewählten Verfahren auf verschiedene Problemgrößen angewandt und deren Lösungsqualität sowie deren benötigte Rechenzeit verglichen.

5.2.1 Ausgewählte Verfahren

Für das Routen der einzelnen Touren sollen die sukzessive-Konstruktionsheuristik, das 2-Opt-Verfahren sowie eine Lin-Kernighan-Implementierung getestet werden. Der sukzessiven Konstruktionsheuristik wurde gegenüber der Nearest-Neighbor-Heuristik nicht nur deshalb der Vorzug gegeben, weil diese bessere Ergebnisse liefert, sondern vielmehr weil damit auch ein Verfahren zur Verfügung steht, mit Hilfe dessen gegebene Touren erweitert werden können, ohne diese neu erstellen zu müssen. Bei der verwendeten Implementierung dieses Verfahrens wird auf eine Vorgabe der Reihenfolge, in der die Knoten in die Tour integriert werden (wie es in [9] vorgeschlagen wird), verzichtet, da es auch zum Erweitern gegebener Touren verwendet wird. Dieses Verfahren soll als Ausgangspunkt der Untersuchungen dienen, d.h. die einfachste und schnellste Möglichkeit der Routenzusammenstellung repräsentieren.

Als erste Steigerung dieser einfachen Konstruktionsheuristik soll ein simpler 2-Opt-Algorithmus untersucht werden. Dieser wurde lediglich als First-Fit-Variante implementiert, d.h. sobald durch den Tausch zweier Kanten eine Verbesserung erzielt werden kann, wird dieser Tausch vollzogen. Kann durch den Tausch zweier Kanten keine Verbesserung mehr erzielt werden, stoppt der Algorithmus. Als Ausgangspunkt kann entweder die Lösung des

sukzessiv konstruierenden Verfahrens oder aber eine zufällig erstellte herangezogen werden. Wie sich die Wahl des Ausgangspunkts auf die Lösungsqualität sowie die Rechenzeit auswirkt, wird im nächsten Abschnitt erörtert.

Der Lin-Kernighan-Algorithmus soll als weitere Steigerung zum 2-opt-Verfahren dienen. Die grundlegende Vorgehensweise dieses Verfahrens wurde bereits in Abschnitt 3.2.2 gezeigt, weshalb auf diese hier nicht weiter eingegangen wird. Zu erwähnen sind hier lediglich die verschiedenen Parameter, deren Auswirkungen auf das erzielte Ergebnis im nächsten Abschnitt untersucht werden. Zunächst kann man, wie auch bei dem 2-Opt-Verfahren, zwischen einem zufällig erstellten und einem sukzessive konstruierten Ausgangspunkt wählen. Weiters kann die Größe der untersuchten Nachbarschaft jedes Knoten eingestellt werden. Dies bedeutet, es werden lediglich Kanten untersucht, die zu einem Knoten führen, der sich innerhalb dieser Nachbarschaft befindet. Als letzter wichtiger Parameter wird natürlich λ übergeben. Dieser Parameter stellt ein, wie viele Kanten auf einmal getauscht werden können, und stellt somit den Parameter für die Suchintensität dar.

5.2.2 Verschiedene Varianten des 2-Opt- sowie des Lin-Kernighan-Verfahrens

In diesem Abschnitt werden verschiedene Einstellungen für die eben erwähnten Parameter der ausgewählten Verfahren verglichen, ehe im nächsten Abschnitt die Verfahren mit den am bestgeeigneten Einstellungen untereinander verglichen werden. Zunächst wird sowohl für das 2-Opt-Verfahren als auch für das Lin-Kernighan-Verfahren der Unterschied zwischen einer zufällig erstellten und einer durch den sukzessive Konstruktionsalgorithmus generierten Startlösung untersucht. Dazu wurden für die Problemgrößen 5-10 Knoten, 10-20 Knoten, 20-30 Knoten sowie 30-50 Knoten jeweils 100 Probleminstanzen aus der vorhandenen Distanzmatrix zufällig erstellt. Diese wurden dann mit den zwei Algorithmen einmal mit einem zufälligen Startpunkt und einmal mit einem sukzessive konstruierten Startpunkt gelöst. Für das Lin-Kernighan-Verfahren wurden natürlich die übrigen Parameter bei beiden Durchgängen gleich eingestellt ($\lambda = 3$ und alle Knoten als potenzielle Tauschpartner).

In Abbildung 18 auf der nächsten Seite sind die Ergebnisse gezeigt, wobei als Indikator für die Lösungsqualität der Unterschied zur Lösung der sukzessiven Konstruktionsheuristik herangezogen wird. Bei Betrachtung dieses Ergebnisses fällt auf, dass der Unterschied zwischen den zwei Varianten beim 2-Opt-Algorithmus wesentlich stärker ausgeprägt ist als beim Lin-Kernighan-Verfahren. So kann beim 2-Opt-Algorithmus durchaus behauptet werden, dass jene Variante mit einem sukzessive-konstruierten Startpunkt tendenziell zu besseren Lösun-

gen führt. Denn bei den Testläufen zu den Problemen mit 10 bis 20 Knoten gibt es zum Beispiel insgesamt 60 TSPs bei denen sich die Ergebnisse der beiden 2-Opt-Varianten unterscheiden. Davon ist bei 52 die Lösung der Sukzessiv-Variante besser, d.h. das Signifikanzniveau liegt bei 0,00000026 %. Selbst bei den Problemen mit weniger als 10 Knoten, bei denen die Ergebnisse noch sehr knapp beieinanderliegen, erreicht das Signifikanzniveau bereits einen Wert unter einem Prozent (22 unterschiedliche Lösungen, davon 17 bessere mit Sukzessiv-Variante). Auch der Vergleich der Laufzeiten dieser beiden 2-Opt-Varianten spricht eine deutliche Sprache: Die Zufalls-Variante benötigt sechs Mal so lange wie die Sukzessiv-Variante (wobei angemerkt werden muss, dass auch 124ms für die Lösung aller 400 Probleme keine sehr lange Rechenzeit ist).

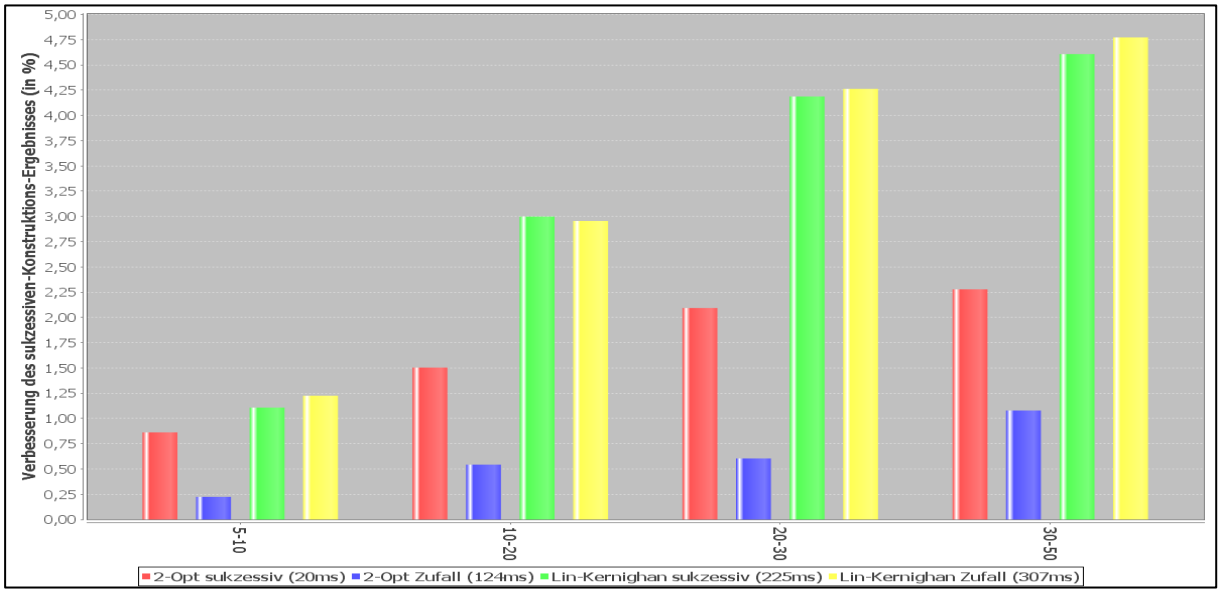


Abbildung 18: Unterschied zwischen zufälliger und sukzessiv-konstruierter Startlösung

Beim Lin-Kernighan-Verfahren kann hinsichtlich der durchschnittlichen Lösungsqualität kein signifikanter Unterschied zwischen den beiden Varianten beobachtet werden. Doch es fällt auf, dass sich bei den größeren Problemen (30-50 Knoten) bereits 87 der erzielten Ergebnisse unterscheiden (40 waren mit zufälligem Startpunkt besser, 47 mit sukzessiv-konstruiertem). Aus diesem Grund wäre es für größere Problemstellungen durchaus sinnvoll, mit beiden Varianten jeweils einen Testlauf durchzuführen. Hinsichtlich der Rechenzeit fällt der Unterschied mit ca. 25% relativ gering aus.

Für den Lin-Kernighan-Algorithmus gibt es neben der Startlösung noch die zwei weiteren Parameter Nachbarschaftsgröße und λ, also die maximale Anzahl an vertauschten Kanten je Iteration. Die Einschränkung der untersuchten Nachbarschaft wirkt sich natürlich positiv auf die Laufzeit aus, jedoch könnte dadurch der Weg zu besseren Lösungen versperrt werden. Da die Laufzeit des Verfahrens für Probleme der vorliegenden Größe ohnehin sehr gering ist,

für die Lösung der 100 Probleme mit 10 bis 20 Knoten wurden insgesamt lediglich 7ms benötigt, werden generell alle Knoten als potenzielle Tauschpartner untersucht. Wie sich eine Veränderung von λ auf die Lösungsqualität sowie die Rechenzeit auswirkt, wird im folgenden Abschnitt näher untersucht.

5.2.3 Analyse der Testergebnisse

Auf Basis der eben gezeigten Untersuchungen werden drei verschiedene Varianten getestet und mit den Ergebnissen der sukzessiven Konstruktion verglichen. Diese sind der 2-Opt-Algorithmus mit sukzessiv-konstruiertem Startpunkt, sowie das Lin-Kernighan-Verfahren mit $\lambda = 3$ sowie mit $\lambda = 4$ (jeweils ein Durchgang mit sukzessiv-konstruiertem Startpunkt und einer mit zufällig erstelltem, keine Einschränkung der Nachbarschaft). Wie bereits im vorigen Abschnitt wurden die Algorithmen für die vier verschiedenen Problemgrößen 5-10 Knoten, 10-20 Knoten, 20-30 Knoten sowie 30-50 Knoten auf 100 verschiedene zufällig generierte Probleminstanzen getestet. Wie zu Beginn dieses Kapitels erwähnt wurde, soll untersucht werden, inwieweit bei den unterschiedlichen Problemgrößen das Ergebnis der sukzessiven Konstruktion durch die verschiedenen Verfahren verbessert werden kann, und welche Rechenzeit dafür in Anspruch genommen wird. Daraus soll abgeleitet werden, welche Verfahren sich für die Verwendung innerhalb der anschließend gezeigten Heuristiken zur Lösung des VRPs eignen.

Bei den TSPs mit fünf bis zehn Knoten können durch den 2-Opt-Algorithmus lediglich 16 Lösungen verbessert werden, wovon die Verbesserung nur bei einem Problem über 3% liegt. Die durchschnittliche Verbesserung liegt bei 0,28%. Mit dem Lin-Kernighan-Verfahren können 29 bzw. 31 Lösungen verbessert werden, wobei bei acht bzw. neun Problemen die Verbesserung über 3% liegt. Auch mit diesem Verfahren liegt bei solch geringer Anzahl an Knoten die durchschnittliche Verbesserung lediglich bei 0,87 bzw. 1,02%.

Wie in Abbildung 19 auf der nächsten Seite ersichtlich ist, fällt die Verbesserung bei Problemen mit 10 bis 20 Knoten schon deutlicher aus. Hier können bereits 60 Lösungen mit Hilfe des 2-Opt-Verfahrens verbessert werden. Davon konnten 18 Lösungen um mehr als 3% verbessert werden, die durchschnittliche Verbesserung liegt bei 1,59%. Das Lin-Kernighan-Verfahren findet bei dieser Problemgröße bereits in 79 bzw. 80 Fällen eine bessere Lösung als die sukzessive Konstruktionsheuristik. Mit einer durchschnittlichen Verbesserung von 3,52 bzw. 3,64% liegt auch hier das Lin-Kernighan-Verfahren deutlich vor dem 2-Opt-Algorithmus, obwohl sich die Rechenzeiten kaum unterscheiden. Diese ist hier mit 44ms für das Lin-Kernighan-Verfahren ($\lambda = 3$) sogar etwas geringer als jene des 2-Opt-Algorithmus

(51ms). Wie schon bei den kleinsten Probleminstanzen, kann auch hier durch die Erhöhung des λ-Wertes keine große Verbesserung erzielt werden.

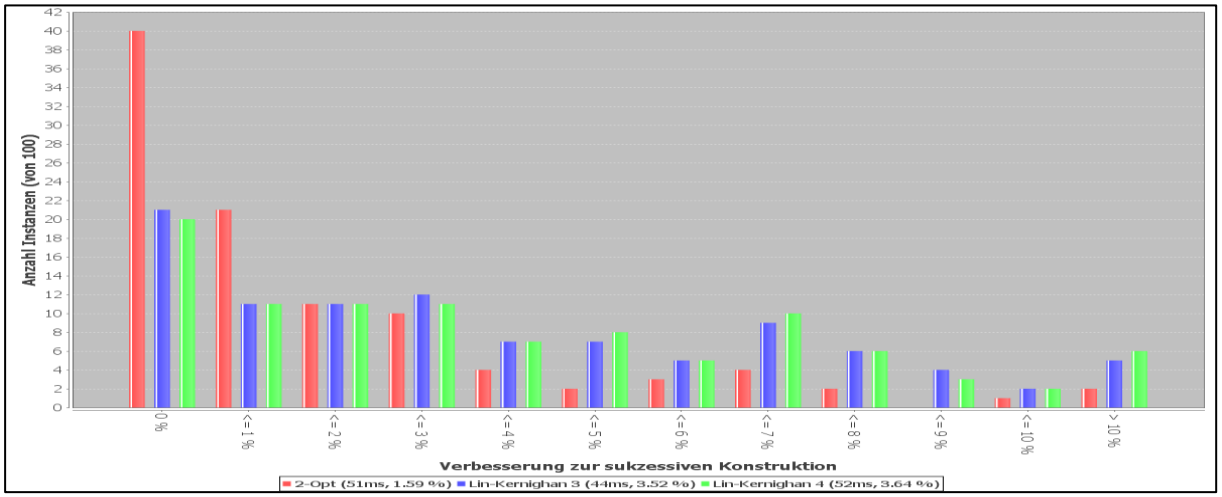


Abbildung 19: Vergleich verschiedener Verfahren für TSPs (10 bis 20 Knoten)

Auch bei den nächstgrößeren Problemen (20 bis 30 Knoten bzw. 30 bis 50 Knoten) ist bezüglich der Lösungsqualität dieselbe Tendenz zu erkennen, wie bei den bisher gezeigten: Je höher die Anzahl an Knoten, desto mehr sukzessiv-konstruierte Lösungen können durch die getesteten Verfahren verbessert werden. Wie in Abbildung 20, die die Ergebnisse für die Probleme mit 30 bis 50 Knoten zeigt, zu sehen ist, kann man hier bereits in 98 bzw. allen 100 Fällen mit Hilfe des Lin-Kernighan-Verfahrens eine Verbesserung erzielen. Auch der Grad der Verbesserung steigt mit der Anzahl an Knoten. So können bei dieser Problemgröße schon 15 bzw. 13 Ergebnisse um mehr als 10% verbessert werden und die durchschnittliche Verbesserung mit dem Lin-Kernighan-Verfahren liegt bei 6,00 bzw. 6,35%. Auch in Bezug auf die Unterschiede zwischen den getesteten Verfahren bleibt die Tendenz die gleiche wie bisher: Das 2-Opt Verfahren ist deutlich schwächer als die beiden Lin-Kernighan-Implementierungen, diese unterscheiden sich untereinander jedoch kaum.

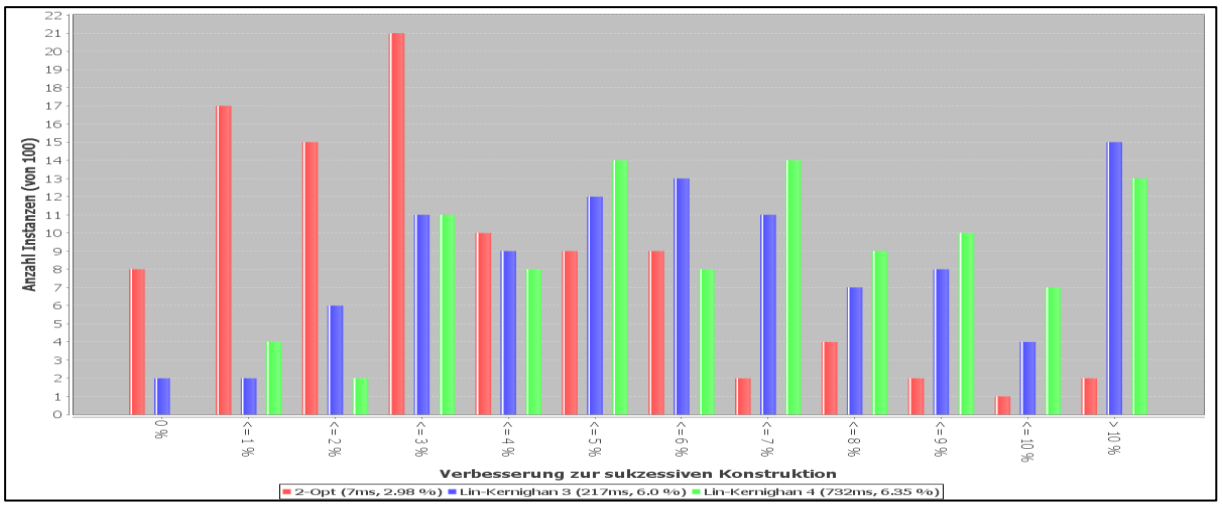


Abbildung 20: Vergleich verschiedener Verfahren für TSPs (30 bis 50 Knoten)

Die Unterschiede der Rechenzeiten wird bei den größeren Problemstellungen immer deutlicher und zeigt die logische Tendenz: Das 2-Opt-Verfahren ist deutlich schneller als die Lin-Kernighan-Implementierungen, von denen jene mit dem höheren λ-Wert mehr Zeit in Anspruch nimmt.

Aus den eben gezeigten Analysen können zwei wesentliche Erkenntnisse gewonnen werden, die für die Entwicklung der im Anschluss erörterten Heuristiken für die Lösung des gesamten Tourenplanungs-Problems von Bedeutung sind: Erstens ist zu erkennen, dass bei kleinen Touren, also Touren mit weniger als 10 Abladestellen, das schnelle sukzessive Konstruktionsverfahren in den meisten Fällen sehr gute Lösungen liefert. Damit eignet es sich vor allem für alle lokalen Suchverfahren und Verbesserungsheuristiken, bei denen für sehr viele verschiedene Zusammenstellungen der Touren die Kosten der dadurch entstehenden Routen (mit möglichst geringem Rechenaufwand) abgeschätzt werden müssen. Die zweite wesentliche Erkenntnis ist, dass sich die Rechenzeiten von dem 2-Opt-Algorithmus und dem Lin-Kernighan-Verfahren erst bei Problemen mit mehr als 20 Abladestellen spürbar unterscheiden – die Lösungsqualität aber auch bei kleineren Problemen schon deutlich für das Lin-Kernighan-Verfahren spricht. Damit ist klar, dass für etwaige Zwischen- bzw. Endoptimierungen der Routen das Lin-Kernighan-Verfahren dem 2-Opt-Algorithmus vorzuziehen ist.

5.3 Konstruktionsheuristiken und lokale Suche

In diesem Abschnitt werden nun die ersten Lösungen für das vorliegende Tourenplanungsproblem konstruiert bzw. in weiterer Folge durch ein lokales Suchverfahren verbessert. Wie bereits im Abschnitt zur Zielfunktion erwähnt, ist die tatsächliche Qualität einer Lösung nur sehr schwer messbar, da nicht abgeschätzt werden kann, wie eine optimale, oder dem Optimum sehr nahe, Lösung aussehen soll. Aus diesem Grund können die hier gezeigten Verfahren natürlich nur hinsichtlich dem Zielerreichungsgrad auf Basis der im Abschnitt 5.1.2 präsentierten Zielfunktion verglichen werden. Um dadurch aber auch wirklich für den praktischen Einsatz verwendbare Tourenpläne zu generieren, werden die generierten Lösungen laufend der Sichtkontrolle unterzogen. Dabei werden das Gesamtbild aller Touren auf der Karte, die Routen der einzelnen Touren, sowie die Gesamt- und Einzelladeraumauslastungen untersucht. Hierbei wird zunächst darauf geachtet, ob es auch in Gebieten mit einer höheren Auftragsdichte langgezogene Touren gibt, sich die Touren häufig überschneiden, oder ob die Routen der Touren sichtliche Mängel aufweisen. Neben diesem Bild des Tourenplans wird natürlich auch der durchschnittlichen Laderaumauslastung eine große Aufmerksamkeit ge-

schenkt. Denn insbesondere wenn man sich Frächtern bedient, spielt diese eine große Rolle, da eine hohe Laderaumauslastung natürlich eine geringere Anzahl an Touren bedeutet. Bei der Kontrolle der Laderaumauslastung werden auch die Auslastungen der einzelnen Touren untersucht, wobei insbesondere darauf geachtet wird, ob es mehrere Touren mit geringer Laderaumauslastung im selben Gebiet gibt. Zusammengefasst kann als Ziel dieser Gesamtkontrolle erklärt werden, dass selbst bei längerer Betrachtung des Tourenplans keine Verbesserungen (hinsichtlich der verwendeten Zielfunktion) erkennbar sein dürfen! Wird ein genereller Mangel in allen Lösungen festgestellt, so muss dieser durch Anpassung der Zielfunktion ausgemerzt werden.

Da sich die Lösungsverfahren natürlich bei verschiedenen Problemstellungen unterschiedlich verhalten (insbesondere was die Rechenzeit betrifft), und in dieser Arbeit primär für die vorliegende reale Problemstellung eine gute Lösung erarbeitet werden soll, müssen die ausgewählten Verfahren natürlich auf möglichst realitätsnahe Probleminstanzen getestet werden. Aus diesem Grund werden alle nachfolgenden Verfahren auf, mit Hilfe der realen Distanzmatrix zufällig erstellten, Probleme mit 200, 300 bzw. 400 Knoten getestet. Diese entspricht in etwa der Bandbreite an wöchentlich belieferten Kunden je Werk. Da aber nicht nur die Anzahl der Knoten, sondern auch die Verteilung der Bedarfe je Kunden von Bedeutung ist, wird auch diese an die tatsächlich vorliegende Verteilung angepasst. Dafür wurden alle belieferten Kunden einer Kalenderwoche anhand deren Bedarfs in sieben verschiedene Kategorien eingeteilt, und anhand dieser Aufteilung werden die Bedarfe bei der Generierung von Testinstanzen zugeteilt. Diese Verteilung unterscheidet sich von einer Gleichverteilung insbesondere durch die starken Ausreißer nach oben (hervorgerufen durch Zentrallager diverser Kunden).

5.3.1 Vergleich zwischen Nearest-Neighbor-Search und dem Savings-Algorithmus

Als die zwei zu untersuchenden Konstruktionsheuristiken wurden die Nearest-Neighbor-Heuristik und der im Abschnitt 3.3 beschriebene Savings-Algorithmus ausgewählt. Der im Abschnitt 3.3 ebenfalls gezeigte Sweep-Algorithmus wird deshalb nicht untersucht, da von diesem auf Grund der Eigenschaften des vorliegenden Graphen keine guten Lösungen zu erwarten sind – die Depots liegen nicht im Zentrum der Kunden und die Entfernungen sind zum Teil sehr groß. Auch wenn der Petal-Algorithmus in der Literatur als Konstruktionsheuristik bezeichnet wird, und somit vermeintlich in diesen Teil der Arbeit gehört, wird auf diesen erst bei der Beschreibung der entwickelten Verbesserungsheuristiken näher eingegangen, da er dort eine entscheidende Rolle spielt.

Bei der Analyse der zwei ausgewählten Verfahren soll insbesondere untersucht werden, wie sich die Rechenzeit verhält, wie sich Veränderungen der Einstellungen auswirken, welche Vor- bzw. Nachteile die beiden Algorithmen aufweisen und natürlich wie gut die generierten Lösungen sind.

Die Nearest-Neighbor-Heuristik zum Erstellen von Tourenplänen wurde folgendermaßen implementiert: Zunächst wird ein Startpunkt für eine Tour gewählt, wobei dem Algorithmus als Parameter übergeben werden kann, ob immer der vom Depot am weitesten entfernte, noch nicht besuchte Knoten genommen werden soll. Nun wird so lange jener noch nicht besuchte Kunde der Tour zugeteilt, der die Kapazitätsrestriktion nicht verletzt und die geringste Distanz zu einer der bereits zugeteilten Abladestellen aufweist. Wird kein solcher Kunde gefunden, wird eine neue Tour begonnen. Wurden alle Kunden einer Tour zugeteilt, werden die Routen mittels dem im vorigen Abschnitt beschriebenen Lin-Kernighan-Verfahrens optimiert.

Bei dem implementierten Savings-Algorithmus werden die Savingswerte für die verschiedenen Tourenpaare nicht nur durch den Distanzunterschied (wie in der einfachsten Form verwendet) repräsentiert, sondern es wird eine temporäre Tour erstellt, die alle Kunden beider Touren enthält, wobei die Konstruktion der Route mit Hilfe des sukzessiven Konstruktionsverfahrens erfolgt. Die Höhe des Savings wird anschließend durch die Differenz der Kosten der beiden ursprünglichen Touren und der Kosten der neuen temporären Tour ausgedrückt. Damit werden sowohl die Fixkosten je Tour als auch der Clusterfaktor berücksichtigt. Nach erfolgter Savings-Berechnung werden die Touren nach der parallelen Vorgangsweise zusammengelegt, d.h. die Savings werden der Höhe nach gereiht, und anschließend wird in absteigender Reihenfolge versucht, durch Zusammenlegen der entsprechenden Touren das dazugehörige Saving zu erzielen. Kann keine Verbesserung mehr durch Zusammenlegen zweier Touren erreicht werden, werden die Routen, wie bei der Nearest-Neighbor-Heuristik, mit Hilfe des Lin-Kernighan-Verfahrens optimiert. Als Parameter kann dem Algorithmus übergeben werden, wie häufig die Savings neu berechnet werden sollen.

Für den Vergleich der Heuristiken werden eine Nearest-Neighbor-Implementierung und drei verschiedene Savings-Varianten herangezogen. Die Nearest-Neighbor-Heuristik wird nur mit jener Variante getestet, bei der immer mit dem am weitesten entfernten Knoten begonnen wird, da diese beim direkten Vergleich – wie zu erwarten war – deutlich bessere Ergebnisse erzielte: Bei 100 verschiedenen 300-Knoten-Problemen war in 99 Fällen das Ergebnis der ersten Variante besser, im Schnitt um ca. 10%! Der Savings-Algorithmus wird mit drei verschiedenen Häufigkeiten der Savings-Neuberechnung untersucht, selten („Savings 1"), mittel („Savings 2") und häufig („Savings 3"). Mit Hilfe dieser vier Algorithmen wurden jeweils 100

verschiedene Probleme mit 200, 300 bzw. 400 Knoten gelöst, und für jedes dieser vier Verfahren die prozentuelle Abweichung von den jeweils besten Ergebnissen errechnet (siehe Abbildung 21).

Bei Betrachtung dieser Ergebnisse fällt sofort auf, dass mit Hilfe des Nearest-Neighbor-Algorithmus deutlich schlechtere Ergebnisse erzielt werden als mit den drei Savings-Varianten. Auffallend ist dabei aber auch, dass sich mit steigender Knotenanzahl dieser Unterschied verringert, von 10% Gesamtabweichung von den besten Ergebnissen bei 200 Knoten auf 6,3% bei Problemen mit 400 Kunden. Die Savings-Varianten weisen bei allen Problemgrößen nur einen sehr geringen Unterschied auf, wobei, wie zu erwarten war, mit der Häufigkeit der Savings-Neuberechnung die Lösungsqualität (leicht) zunimmt.

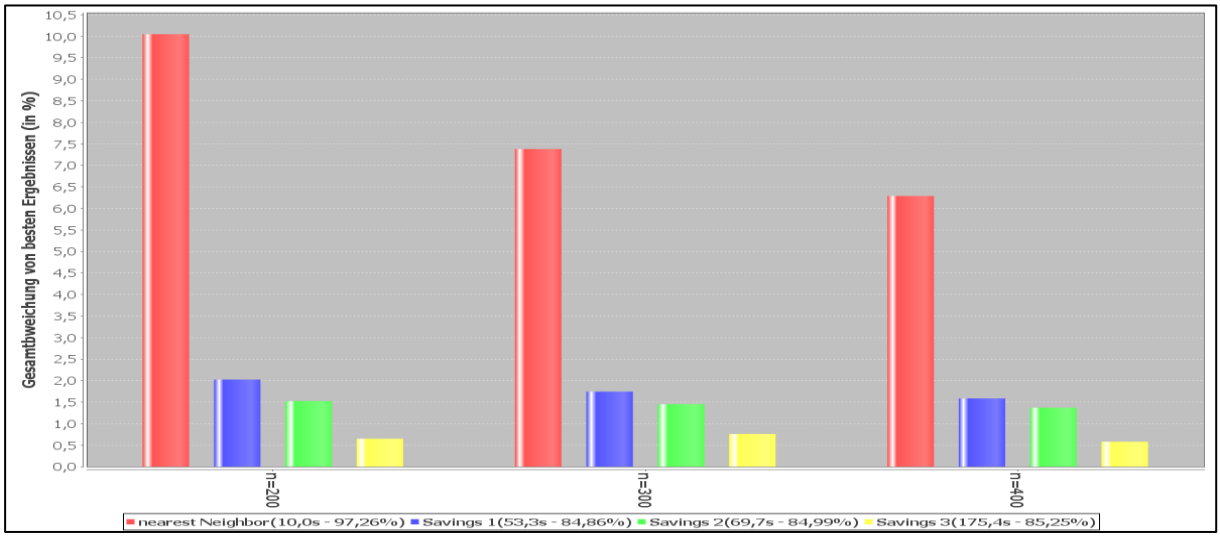


Abbildung 21: Nearest-Neighbor vs. Savings: Vergleich der Zielfunktionswerte

Einen deutlichen Unterschied gibt es zwischen dem Nearest-Neighbor-Algorithmus und dem Savings-Verfahren auch in der durchschnittlichen Laderaumauslastung (Prozentangabe in der Legende des Diagramms in Abbildung 21). So wird bei ersterem eine durchschnittliche Auslastung von über 97% erreicht, während dieser Wert bei den Savings-Implementierungen bei ca. 85% liegt. Diese hohe Auslastung beim Nearest-Neighbor-Algorithmus kommt daher, dass dieser eine Tour so lange um einen weiteren Knoten erweitert, bis keiner mehr ohne Verletzung der Kapazitätsrestriktion hinzugefügt werden kann (ohne Berücksichtigung der in Kauf genommenen Zusatzkosten).

Die Auswertung der Rechenzeiten zeigt ein durchaus erfreuliches Ergebnis: Beim Nearest-Neighbor-Algorithmus wurden lediglich 10 Sekunden für das Lösen aller 300 Probleme benötigt, und auch die rechenzeitintensivste Savings-Implementierung war mit dem Lösen aller Probleme nur 175 Sekunden beschäftigt. Dies entspricht einer durchschnittlichen Rechenzeit von etwa 0,6 Sekunden je Problem. Zu sehen ist auch, dass beim Savings-Algorithmus die

Berechnung der Savings wesentlich mehr Zeit in Anspruch nimmt, als die Zusammenlegung der Touren. Aus diesem Grund benötigt die Savings-Implementierung mit einer seltenen Neuberechnung der Savings gesamt nur 53 Sekunden, also durchschnittlich ca. 0,2 Sekunden je Problem.

Bei Betrachtung der Tourenpläne ist der Qualitätsunterschied zwischen dem Nearest-Neighbor-Verfahren und dem Savings-Algorithmus ebenfalls deutlich sichtbar. Wie in Abbildung 22, die die Lösungen dieser beiden Verfahren für eines der 400-Knoten-Probleme zeigt, zu sehen ist, weist die Nearest-Neighbor-Lösung ein deutliches „Durcheinander" auf, während bei der Savings-Lösung auf den ersten Blick keine großen Mängel erkennbar sind. Der Zielfunktionswert liegt bei diesen Lösungen um 6,3% auseinander, wobei die erste Lösung mit 37 Fahrzeugen (Laderaumauslastung: 96,9%) und die zweite mit 42 (Laderaumauslastung: 85,4%) auskommt. Die gesamt zurückgelegte Distanz ist bei diesem Beispiel bei der Savings-Lösung um 7,5% geringer.

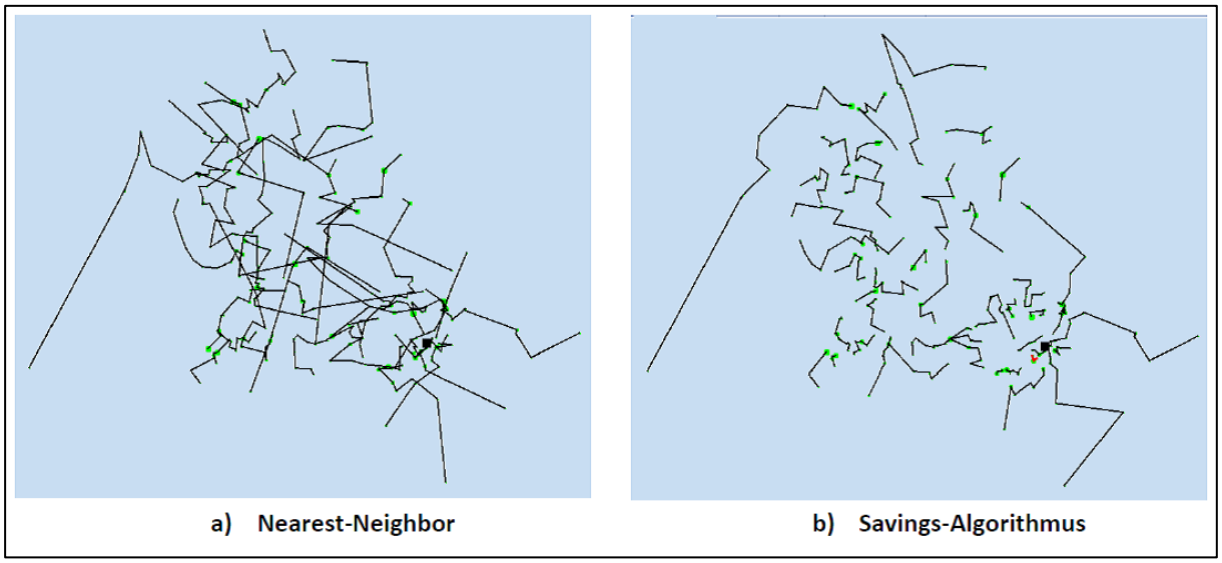


Abbildung 22: Nearest-Neighbor vs. Savings: Vergleich des Tourenplans

Das oben gezeigte und beschriebene Bild ist repräsentativ für die gesamte Testreihe: Während mit Hilfe des Savings-Algorithmus größtenteils durchaus ansprechende Lösungen generiert werden, werden die durch das Nearest-Neighbor-Verfahren erzeugten Lösungen im praktischen Einsatz sicherlich niemanden überzeugen können. Einzig und allein die etwas gering erscheinende Laderaumauslastung der Savings-Lösungen kann als offensichtliches Manko dieses Verfahrens angesehen werden.

5.3.2 Lokale Suche mit Hilfe von *λ-interchanges*

Da nun die ersten, teils ganz passablen, Lösungen für das vorliegende Problem generiert wurden, wird an dieser Stelle untersucht, wie diese Ergebnisse durch ein lokales Suchverfah-

ren verbessert werden können. Für die Generierung von Nachbarschaftslösungen wird auf die λ-interchanges-Definition von Osman [30], die in Abschnitt 3.4.1 beschrieben ist, zurückgegriffen. Bei der Ausführung dieses Suchverfahrens gibt es verschiedene Einstellungsmöglichkeiten, deren Auswirkungen auf das erzielte Ergebnis (bzw. auf die Laufzeit) analysiert werden sollen.

Zunächst kann natürlich das λ als Parameter übergeben werden. Dieser Parameter definiert wie viele Knoten innerhalb einer Iteration maximal von einer Tour in eine andere verschoben werden können. Weiters kann, wie eigentlich bei jedem lokalen Suchverfahren, zwischen einer First-Fit- und einer Best-Fit-Variante gewählt werden. Als letzte wichtige Einstellungsmöglichkeit ist noch die Begrenzung der zu untersuchenden Nachbarschaft zu erwähnen: Für jeden Knoten gibt es eine Liste, die alle anderen Knoten nach aufsteigender Distanz zu diesem enthält. Nun wird dem Algorithmus ein Faktor zwischen 0 und 1 übergeben, der jenen Anteil dieser Liste definiert, der als Nachbarschaft anerkannt wird. Das bedeutet, das Verschieben eines Knoten wird nur dann geprüft, wenn zumindest einer der Knoten der in Frage kommenden Tour innerhalb dieser Nachbarschaft liegt. Dadurch soll natürlich versucht werden, die Laufzeit zu beschränken, ohne einen zu großen Verlust bei der Lösungsqualität hinnehmen zu müssen. Da bisher zwei Konstruktionsheuristiken zur Verfügung stehen, die sehr unterschiedliche Lösungen generieren, soll auch verglichen werden, wie sich die Wahl der Startlösung auf das erzielte Ergebnis auswirkt.

Die hier gezeigten Verfahren werden auf dieselben Probleminstanzen getestet, wie auch bereits vorher die Konstruktionsheuristiken, das heißt auf jeweils 100 Probleme mit 200, 300 bzw. 400 Knoten. Als Vergleichskriterium wird die Verbesserung zu den Lösungen des durchschnittlichen Savings-Algorithmus herangezogen. Es wurden vier verschiedene Einstellungen gewählt, die jeweils mit einer Savings-Ausgangslösung und einer Nearest-Neighbor-Ausgangslösung getestet werden: „First-Fit" und „Best-Fit" (jeweils mit λ=1 und 0.1 als Nachbarschaftsfaktor), „Weit" (Best-Fit-Variante mit λ=1 und 0.25 als Nachbarschaftsfaktor) sowie „Lamda 2" (Best-Fit-Variante mit λ=2 und 0.1 als Nachbarschaftsfaktor). Die Ergebnisse zu diesen Einstellungen sind in Abbildung 23 auf der nächsten Seite zu sehen.

Bei Betrachtung dieses Diagramms fällt als erstes auf, dass mit Hilfe von Nearest-Neighbor-Ausgangslösungen deutlich bessere Ergebnisse erzielt werden, als mit Savings-Ausgangslösungen. Dies erscheint auf Grund der Analysen des vorigen Abschnitts, bei denen die Savings-Konstruktionsheuristik als eindeutiger Sieger hervorging, zunächst merkwürdig, doch bei genauerer Betrachtung findet man einen wesentlichen Grund dafür, die Laderaumauslastung. Diese kann durch das lokale Suchverfahren nicht verbessert werden und

liegt somit bei den Savings-Varianten weiterhin bei etwa 85%, während diese für Nearest-Neighbor-Lösungen bei ca. 97% bleibt. Dies bedeutet, die lokale Suche wirkt hauptsächlich (bzw. ausschließlich) auf die Gesamtdistanz sowie den Clusterfaktor, und diesbezüglich scheint der reine Savings-Konstruktionsalgorithmus bereits nahe an ein lokales Optimum heranzukommen. Aus diesem Grund wird auch für die Verbesserung dieser Lösungen weniger Zeit benötigt, als für die Verbesserung der Nearest-Neighbor-Lösungen aufgewendet werden muss.

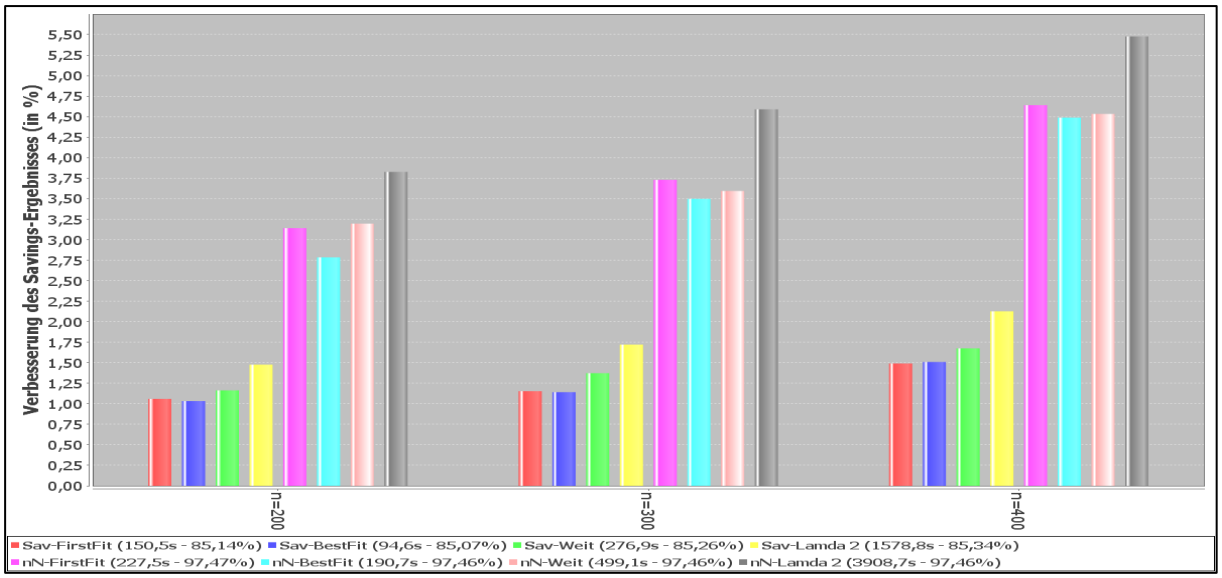


Abbildung 23: lokale Suche mittels λ-interchanges - Vergleich verschiedener Parameter

Anhand dieser Analyse ist auch zu erkennen, dass es zwischen den First-Fit und den Best-Fit-Varianten keine signifikanten Unterschiede gibt, auch die Erweiterung der Knoten-Nachbarschaft bringt nur eine geringe Verbesserung. Diese muss aber mit einer knappen Verdreifachung der Rechenzeit erkauft werden. Einen deutlicheren Sprung gibt es bei der Erhöhung des Lamda von 1 auf 2, doch dies wirkt sich noch intensiver auf die Rechenzeit aus. So werden zur Lösung aller 300 Probleme mit der Best-Fit Nearest-Neighbor-Variante lediglich 190,7 Sekunden benötigt, während mit der „Lamda 2"-Einstellung bereits 3908,7 Sekunden gerechnet wird. Dies bedeutet für ein einzelnes Problem die Steigerung von ca. 0.6 Sekunden auf in etwa 13 Sekunden. Ein Testlauf mit λ=3 für die 100 Probleme mit 400 Knoten benötigt bereits über 7,5 Stunden also ca. 4,5 Minuten je Problem. Die erzielte Verbesserung zum Savings-Konstruktionsergebnisses liegt dabei bei 5,1%, anstatt bei 4,7% mit λ=2.

Bei der Durchsicht der konstruierten Tourenpläne kann nun deutlich erkannt werden, dass das „Durcheinander", das bei reinen Nearest-Neighbor-Lösungen sehr stark ausgeprägt ist, durch das lokale Suchverfahren deutlich entwirrt werden kann (was somit auch den niedrigeren Zielfunktionswert rechtfertigt). Wenn man sich zum Beispiel die Tourenpläne in Abbil-

dung 24 ansieht, ist das Bild des Savings-Ergebnisses zwar eventuell etwas „ruhiger", doch bei genauerer Betrachtung der Ergebnisse, ist die bessere Bewertung des linken Tourenplans durchaus gerechtfertigt. Denn erstens werden für die erste Lösung nur 28 anstatt 32 Fahrzeuge benötigt, und zweitens ist die Gesamtdistanz mit 22.500km auch um ca. 500km geringer als mit der Savings-Startlösung.

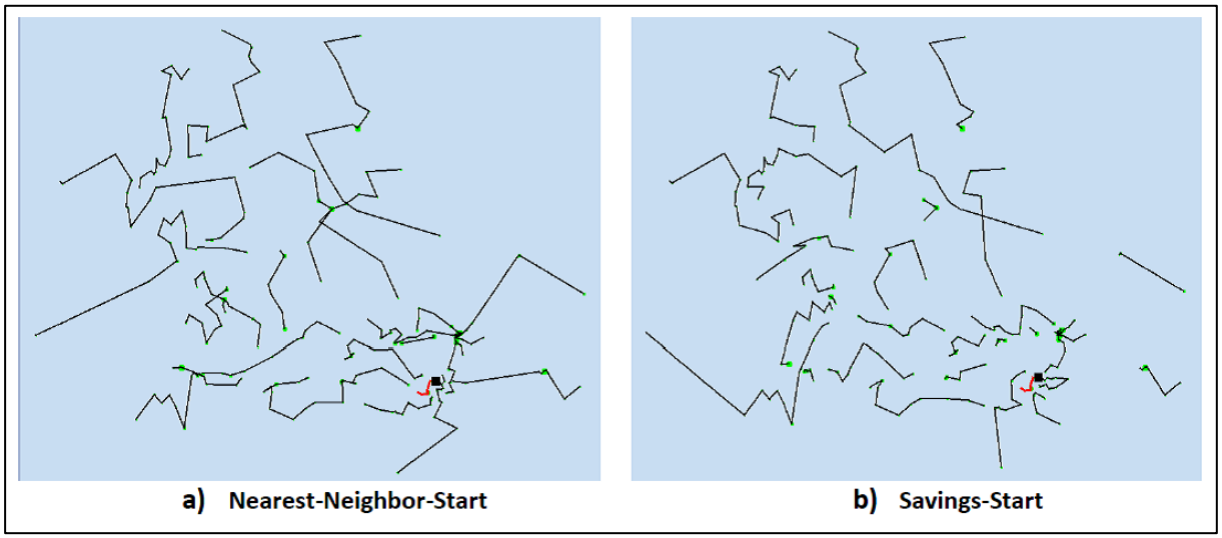


Abbildung 24: Tourenpläne von verschiedenen Startlösungen für die lokale Suche

Die bisher erzielten Ergebnisse zeigen deutlich, dass bereits mit relativ einfachen Heuristiken bzw. lokalen Suchverfahren Lösungen für reale Tourenplanungsprobleme erzielt werden, die durchaus eine Anwendung in der Praxis finden könnten. Auch die Rechenzeit ist bisher mehr als zufriedenstellend, denn mit einer Rechenzeit von etwa 0.6 Sekunden mit $\lambda=1$ liegt man mehr als deutlich noch in dem Bereich, in dem ein manuelles Anstoßen aus dem Tourenplanungsprogramm heraus möglich ist. Auch eine Rechenzeit von ca. 13 Sekunden mit $\lambda=2$ (bei der Nearest-Neighbor-Startlösung) stellt für ein komfortables Interagieren mit dem Programm kein großes Hindernis dar. Bei diesen kurzen Rechenzeiten können während der Tourenplanung mehrere Szenarien getestet werden, wodurch der Tourenplaner selbst auf das erzielte Ergebnis Einfluss nehmen kann. Größtes Manko bisher ist jedoch sicher, dass bisher lediglich eine Variante zur Verfügung steht, bei der eine nahezu maximale Laderaumauslastung vorliegt, und eine, bei der diese mit 85% eher gering erscheint. Hier fehlt die Möglichkeit, vielversprechende Lösungen dazwischen zu generieren bzw. zu testen. Weiters wurde bisher auch nicht die Möglichkeit berücksichtigt, unterschiedliche Fahrzeugtypen zu verwenden. Aus diesem Grund sollen im nächsten Kapitel Verfahren entwickelt und untersucht werden, mit Hilfe derer auch in diesem Bereich des Lösungsraums nach vielversprechenden Tourenplänen gesucht wird.

5.4 Entwicklung geeigneter Verbesserungsheuristiken

Auch wenn die bisher erzielten Ergebnisse schon sehr vielversprechend für den angestrebten Einsatz innerhalb des Tourenplanungs-Prozesses erscheinen, soll innerhalb dieses Kapitels nach Möglichkeiten gesucht werden, noch bessere Lösungen zu finden, bzw. den vorhandenen Lösungsraum noch effizienter zu untersuchen. Dabei soll insbesondere das im vorigen Abschnitt erwähnte Problem mit der Generierung von „ausgewogeneren" Lösungen, d.h. Lösungen, bei denen ein besseres Mittelmaß zwischen hoher Laderaumauslastung und kurzen Distanzen bzw. starker Clusterung gefunden wird, ausgemerzt werden. Dabei wird aber weiterhin der Rechenzeit eine große Aufmerksamkeit geschenkt, denn nur bei einer möglichst kurzen Rechenzeit ist ein komfortables Interagieren mit dem Programm möglich, und somit ein umfassende effiziente Tourenplanung realisierbar. Denn je kürzer die Rechenzeit, desto eher wird der Tourenplaner die Möglichkeit der mehrfachen Testläufe nutzen (und somit mehrere Szenarien testen), wodurch die Wahrscheinlichkeit steigt, einem tatsächlichen (und nicht nur theoretischen) Optimum der Tourenzusammenstellung sehr nahe zu kommen.

5.4.1 Implementierung des genetischen Algorithmus

Als erstes Verbesserungsverfahren soll ein genetischer Algorithmus getestet werden. Als Codierung wird bei der implementierten Form dieses Verfahrens eine Permutation aller Kunden ohne Depot bzw. anderem Separator gewählt. Die Aufteilung der Kunden auf die verschiedenen Touren wird erst bei der Fitness-Berechnung mittels dem in Abschnitt 3.3.3 beschriebenen Petal-Algorithmus durchgeführt. Das heißt, hier macht man sich zu Nutze, dass dieser Algorithmus für eine gegebene Reihenfolge der Knoten, die bestmögliche Einteilung in Touren errechnet (Die detailliertere Beschreibung der hier verwendeten Implementierung dieses Verfahrens folgt in Abschnitt 5.4.3). Durch diese Vorgehensweise muss man sich beim Erstellen von Individuen, bzw. bei der Kreuzung zweier Individuen, keinerlei Gedanken über die Zulässigkeit der durch das Individuum repräsentierten Lösung machen. Da somit jede Permutation der Knoten zulässig ist, können alle bekannten Kreuzungsmethoden, die für TSPs verwendet werden, auch hier zum Einsatz kommen. Eine diesem Prinzip folgende Implementierung des genetischen Algorithmus für das VRP wird auch in [33] beschrieben.

Damit bereits in der Ausgangspopulation gute Individuen vorhanden sind, können die Ergebnisse der vorher beschriebenen Konstruktionsheuristiken durch Zusammenführen der Touren zu einer Gesamtroute („Giant-Route") verwendet werden. Für das Herzstück des genetischen Algorithmus, also der Schleife „Paarungspool füllen, Individuen kreuzen und Population er-

neuern", gibt es zwei verschiedene Abbruchkriterien: Erstens kann die Schleife dadurch unterbrochen werden, dass das beste Ergebnis durch eine bestimmte Anzahl an Iterationen nicht mehr verbessert werden kann. Die zweite Möglichkeit ist, dass die Gesamtanzahl an innerhalb des Kreuzungsprozesses generierten Individuen, die einem in der Population bereits enthaltenen gleicht, einen Grenzwert überschreitet (wobei die Gleichheit der Einfachheit halber nur über den Fitnesswert ermittelt wird). Die Ergebnisse verschiedener Testläufe haben gezeigt, dass diese Überprüfung der Gleichheit der Individuen einen sehr guten Indikator für die Stagnation des Verfahrens darstellt, und somit mit diesem Abbruchkriterium die Anzahl an unproduktiven Iterationen (und damit die Rechenzeit) verringert werden kann. Diese Überprüfung wird auch herangezogen, um den „richtigen" Zeitpunkt für eine teilweise Erneuerung der Population zu erkennen. Dies bedeutet, sobald eine Stagnation des Verfahrens erkannt wird, wird ein Teil der Population durch neue Individuen ersetzt, um so neues „Genmaterial" in den Kreuzungsprozess zu bringen.

Für die Füllung des Paarungspools stehen die zufällige Auswahl („Random Selection", kein Selektionsdruck), die binäre Turnierauswahl („Tournament Selection", geringer Selektionsdruck), die rangbasierende Auswahl („rank-based Selection", mittlerer Selektionsdruck) und die fitnessproportionale Auswahl („Roulette-Wheel-Selection", hoher Selektionsdruck) zur Auswahl. Die Gesamtgröße der Population sowie die Größe des Paarungspools, die ebenfalls Auswirkungen auf das Stagnationsverhalten des Verfahrens haben, sind natürlich ebenfalls frei parametrisierbar. Die Kreuzung der selektieren Individuen wurde auf drei verschiedene Arten implementiert: Die erste Art ist das bekannte „Order-Crossover", wobei ein zufällig ausgewähltes Teilstück der DNA von einem Elternteil in das Kind übertragen, und der Rest der Kunden in der Reihenfolge der DNA des anderen Elternteils aufgefüllt, wird. Die zweite Möglichkeit ist das ebenfalls bekannte „Cycle-Crossover"-Verfahren, bei dem ein (in der Regel nicht zusammenhängendes) Teilstück der DNA gesucht wird, das bei beiden Elternteilen dieselben Knoten besitzt (die Größe des Teilstücks wird bei jedem Kreuzungsprozess vom Verfahren selbst bestimmt). Die Knoten dieses Teilstücks werden nun in der Reihenfolge eines Elternteils übernommen, und der Rest in der Reihenfolge des anderen aufgefüllt. Für genaue Beschreibungen dieser Kreuzungsverfahren sowie der erwähnten Selektionsverfahren sei unter anderem auf [29] verwiesen.

Das letzte implementierte Kreuzungsverfahren ist vom Prinzip her eine Erweiterung des Order-Crossover-Verfahrens: Mit Hilfe einer Zufallszahl zwischen zwei (als Parameter übergebenen) Schranken wird die Länge eines Teilstücks bestimmt, die von der DNA des ersten Elternteils übernommen wird. Anschließend wird eine Anzahl an Knoten bestimmt, die nun in

der Reihenfolge des anderen Elternteils (begonnen von der Position des zuletzt hinzugefügten Knotens) übernommen werden. Dabei müssen natürlich jene Knoten ausgelassen werden, die in der DNA des Kindes bereits vorhanden sind. Dieser Vorgang wird so lange wiederholt, bis alle Knoten übernommen wurden. Durch die Vorgabe der minimalen bzw. maximalen Länge der von einem Elternteil zu übernehmenden Teilstücke, kann bestimmt werden, inwiefern Muster erhalten bleiben sollen. Hier besteht auch die Möglichkeit, im Laufe des Verfahrens die Stranglängen sukzessive zu erhöhen, um so den Lösungsraum zunächst möglichst großflächig zu untersuchen und dann die Suche durch das stärkere Erhalten von Mustern auf vielversprechende Gebiete einzuengen. Zusätzlich zu den drei Kreuzungsverfahren gibt es eine Mutationsmethode, bei der eine zufällig bestimmte Anzahl an Knoten verschoben wird, und eine lokale Suchmethode, bei der die Reihenfolge der Knoten mittels der bereits beschriebenen λ-Interchanges-Methode verbessert wird. Die Wahrscheinlichkeit, dass ein Individuum vor der Berechnung der Fitness einer dieser Methoden übergeben wird, wird durch (für beide Methoden separate) Parameter bestimmt.

Für den letzten wichtigen Schritt des Verfahrens, der Populations-Aktualisierung, stehen ebenfalls mehrere Möglichkeiten zu Auswahl: Die erste ist die zufällige Aktualisierung, bei der zunächst alle Kinder in die Gesamtpopulation übernommen werden, und anschließend die Anzahl an neuen Individuen wieder zufällig entfernt wird (damit die Populationsgröße immer gleich bleibt). Bei Auswahl dieses Verfahrens kommt der Selektionsdruck natürlich ausschließlich über das Füllen des Paarungspools zustande. Als zweite Möglichkeit steht der Individuen-Austausch zur Auswahl. Bei diesem Verfahren wird eine definierte Anzahl an Individuen aus der neuen Generation genommen, und durch diese werden, aus der schlechteren Hälfte der Gesamtpopulation zufällig gewählte, Individuen ersetzt. Eine weitere implementierte Art der Populationsaktualisierung ist der Austausch der Individuen des Paarungspools durch die neue Generation (bzw. wenn die Anzahl an Kindern größer als die Anzahl an Individuen im Paarungspool ist, durch einen Teil davon). Diese Variante sollte natürlich nur gewählt werden, wenn das Kreuzungsverfahren die Muster der Eltern nicht zu sehr zerstört, denn sonst führt diese Populations-Aktualisierung möglicherweise zu einem negativen Selektionsdruck (gute Individuen kommen in den Paarungspool und werden ersetzt). Eine weitere Möglichkeit der Populationsaktualisierung ist es, die Individuen des Paarungspools zunächst aus der Gesamtpopulation zu entfernen, anschließend mit der neuen Generation zu vermischen, und aus diesem Pool, mittels einer rangbasierenden Selektion, die Population wieder aufzufüllen. Damit ist die Wahrscheinlichkeit, dass die Eltern durch ihre Kinder ersetzt werden, abhängig vom Kreuzungserfolg. Die letzte implementierte Möglichkeit der Populationsaktualisierung ist es, zunächst die gesamte neue Generation in die Population aufzunehmen,

und anschließend die Anzahl an aufgenommen Individuen mittels einer (negativen) rangbasierenden Selektion wieder zu entfernen. Für alle Varianten gilt, dass es zu keinem Zeitpunkt des Verfahrens zwei idente Individuen in der Population geben darf (ermittelt über die Gleichheit der Fitness).

Die ersten Tests des genetischen Algorithmus haben gezeigt, dass dieses Verfahren für den gewünschten interaktiven Einsatz in dem Tourenplanungsprogramm auf Grund der langen Rechenzeiten sicherlich nicht geeignet ist. Denn diese liegt selbst bei kleineren Problemen und hohem Selektionsdruck (Abbruch nach 500 bis 2000 Iterationen) schon meist über einer Minute, und bei größeren Problemen schon im zweistelligen Minutenbereich. Aus diesem Grund wurde dieses Verfahren nicht näher untersucht.

Für Problemstellungen, bei denen erstens die Kostenfunktion genau bekannt ist, und zweitens die Rechenzeit eine untergeordnete Rolle spielt, sollte dieses Verfahren jedoch sicherlich als Alternative in Betracht gezogen werden. Es konnten zwar keine wirklichen Verbesserungen der bisher bekannten Ergebnisse erzielt werden, doch dies ist wohl auf die verwendete Parametrisierung, für die auf Grund der relativ geringen Anzahl an Testläufen keine optimale Einstellung gefunden werden konnte, zurückzuführen. Hier hat sich gezeigt, dass es sehr schwer ist, eine Ausgewogenheit zwischen Exploration und Stagnation zu finden, wobei aufgefallen ist, dass bereits ein geringer Selektionsdruck zu einer sehr raschen Stagnation führt. Auch die geeignete Wahl der Kreuzung scheint schwierig, denn werden die Muster nur geringfügig geändert, führt dies eventuell zu keiner Veränderung der daraus resultierenden Lösung, eine zu große Veränderung der Muster lässt das Verfahren jedoch natürlich zu einer Zufallssuche verkommen. Ein schönes Stagnationsverhalten (und auch relativ gute Lösungen) werden mit einem zufälligen Füllen des Paarungspools, der beschriebenen Erweiterung des Order-Crossovers als Kreuzungsverfahrens und der Entfernung der Individuen des Paarungspools aus der Population und anschließendem rangbasierenden Auffüllen durch Eltern bzw. Kinder erzielt. In Abbildung 25 auf der nächsten Seite sind zum Beispiel der Verlauf der durchschnittlichen Kosten der Gesamtpopulation sowie die Entwicklung der Gesamtanzahl an verschiedenen Kanten innerhalb aller, der durch die Individuen repräsentierten, Lösungen für ein 150 Knoten Problem zu sehen, bei dem diese Einstellungen vorgenommen wurden. Anhand dieser Bilder ist deutlich zu erkennen, dass die Suche zunächst zwar sehr weitläufig ist (was an der annähernd gleichbleibenden hohen Anzahl an verschiedenen Kanten zu erkennen ist), sehr schlechte Individuen jedoch schnell aus der Population entfernt werden (zu erkennen an der schnellen Reduktion der Durchschnittskosten zu Beginn der Suche). Erst nach etwa 2000 Iterationen beginnt der Stagnationsprozess (Abfall der Anzahl an verschie-

denen Kanten) und nach 4511 Iterationen wird die Suche auf Grund der Überschreitung der maximalen Anzahl an produzierten Individuen, die ident mit bereits in der Population enthaltenen sind, abgebrochen. Bei dieser Suche konnte trotz 3,5 Minuten Rechenzeit das Savingsergebnis jedoch lediglich um 1,2% verbessert werden.

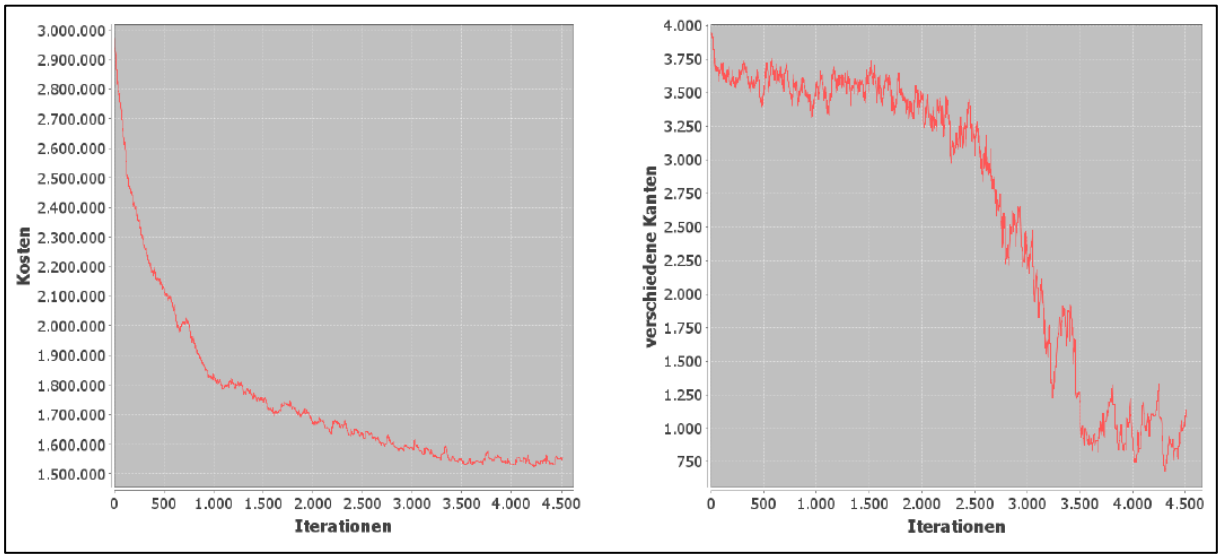


Abbildung 25: Stagnationsverhalten des genetischen Algorithmus

5.4.2 Generierung von Lösungen mittels einem Ameisenalgorithmus

Als zweites Verbesserungsverfahren wird eine Implementierung des Ameisenalgorithmus versucht. Hier wäre es grundsätzlich auch möglich, die Idee umzusetzen, durch die Ameisen nur eine Reihenfolge der Knoten zu generieren, und erst anschließend durch den Petal-Algorithmus die Lösung zu bewerten, und das Pheromon-Update entsprechend durchzuführen. Da dies jedoch sicherlich auch wieder zu sehr hohen Rechenzeiten führen würde, muss ein anderer Weg gefunden werden.

Da es innerhalb des eigentlichen Ameisenalgorithmus schwer ist, eine geeignete Anzahl an Fahrzeugen zu bestimmen, suchen mehrere Ameisenkolonien parallel nach Lösungen, wobei jeder eine fixe Anzahl an zu verwendenden Fahrzeuge zugeordnet wird. Dabei wird die minimale Anzahl durch den Nearest-Neighbor-Algorithmus vorgebeben, die maximale durch den Savings-Algorithmus. Jede dieser Kolonien besteht aus einer definierten Anzahl an Ameisen-Teams, die wiederum aus so vielen Ameisen bestehen, wie Fahrzeuge verwendet werden sollen. In jeder Iteration suchen diese Teams nach Lösungen, wobei die einzelnen Ameisen Touren repräsentieren, und somit ein Ameisen-Team einen gesamten Tourenplan darstellt.

Die Ameisen der Teams suchen parallel, das heißt die Touren werden sukzessive mit Knoten gefüllt, bis alle Knoten zugeordnet sind. Welche Ameise als nächstes einen Knoten aufnimmt, wird über eine Turnierauswahl bestimmt, wobei Ameisen bevorzugt werden, die entweder weit vom Depot entfernt sind, oder noch viel freien Laderaum zur Verfügung haben. Die Auswahl des aufzunehmenden Knotens wird dann mit Hilfe der pseudozufälligen Proportionalitätsregel (siehe Abschnitt 3.4.4) bestimmt, wobei jedoch nur eine eingeschränkte Nachbarschaft des, der Ameise zuletzt hinzugefügten, Knotens berücksichtigt wird. Liegt in dieser Nachbarschaft kein zulässiger Knoten, also wurden diese bereits zugeteilt oder würden sie die Kapazitätsrestriktion verletzen, wird die Ameise als „fertig" markiert, und kann nicht mehr ausgewählt werden.

Wurden alle Ameisen als „fertig" markiert bevor alle Kunden zugeteilt wurden, werden die übrigen mittels einer einfachen Heuristik auf die Ameisen aufgeteilt. Dabei wird für jeden noch nicht zugeteilten Knoten aus den Ameisen, die noch genügend freien Laderaum aufweisen, diejenige ausgewählt, die den nächstgelegenen Knoten besitzt. Kann auch mit Hilfe dieser Heuristik ein Knoten nicht zugeteilt werden, so wird die gesamte Lösung des Teams in der aktuellen Iteration als ungültig deklariert. Für alle gültigen Teams werden die durch sie repräsentierten Tourenpläne bewertet, das heißt die durch die Ameisen dargestellten Touren mittels sukzessiver Konstruktion erstellt, gegebenenfalls durch den Lin-Kernighan-Algorithmus optimiert, und die Kosten dafür berechnet. Beim Pheromon-Update wird zunächst auf allen Kanten der Pheromon-Gehalt verringert (Verdunstung), und anschließend auf den genutzten Kanten des iterationsbesten, oder des bisher besten, Teams erhöht, wobei die Reduktion prozentuell erfolgt und die Erhöhung abhängig von der Lösungsqualität ist. Vor dieser Erhöhung des Pheromon-Gehalts wird die Lösung des iterationsbesten Teams noch mit einer definierten Wahrscheinlichkeit mit der im vorigen Kapitel beschriebenen lokalen Suche optimiert. Die gesamte Suche für jede Kolonie wird nach einer bestimmten Anzahl an Iterationen abgebrochen.

Es wurde bisher nur darauf eingegangen, wie bei dem parallelen Auffüllen aller Touren, diejenige ausgewählt wird, die als nächstes einen Knoten aufnimmt. Ein wesentlicher Bestandteil des implementierten Verfahrens, nämlich wie den verschiedenen Ameisen eines Teams der erste Knoten, also der Startpunkt, zugeteilt wird, wurde jedoch noch nicht gezeigt. Dies ist entscheidend für die Qualität der gefundenen Lösung, denn wenn beispielsweise mehrere Ameisen im selben Gebiet starten, so wird ein Großteil von ihnen große Entfernungen in Kauf nehmen müssen um voll zu werden. Um nun die Startpunkte der einzelnen Ameisen vernünftig über dem gesamten Graphen zu verteilen, wird jedem Team zu Beginn des Verfahrens

eine Cluster-Einteilung der Knoten übergeben (wobei diese Einteilung so viele Cluster besitzt, wie das Team Ameisen). Diese Einteilung wird durch ein einfaches Verfahren ermittelt: Zu Beginn werden so viele Knoten zufällig als Clustermittelpunkte bestimmt, wie Cluster benötigt werden. Anschließend werden die übrigen Knoten demjenigen Cluster zugeordnet, zu dessen Mittelpunkt die (euklidische) Distanz am geringsten ist. Dabei dürfen jedoch die Bedarfe der einem Cluster zugeordneten Knoten einen gewissen Grenzwert nicht überschreiten, wobei dieser Grenzwert als Vielfaches des durchschnittlichen Bedarfs je Cluster definiert wird (hier scheint ein Wert zwischen 1,2 bis 1,5 ausreichend zu sein). Nachdem alle Knoten einem Cluster zugeordnet wurden, wird der Mittelpunkt jedes Clusters (als Schwerpunkt der Knoten) neu berechnet, und alle Knoten erneut dem nächstgelegen Cluster zugeordnet. Dieser Vorgang wird so lange wiederholt, bis keine Verschiebung der Mittelpunkte mehr stattfindet, oder eine gewisse Anzahl an Iterationen durchlaufen wurde (meist stellt sich jedoch nach ein paar Iterationen bereits ein Gleichgewicht ein).

Als Startpunkte für die verschiedenen Ameisen eines Teams werden nun aus jedem Cluster jeweils ein Knoten zufällig ausgewählt, wodurch eine Verteilung der Ameisen über den gesamten Graphen gewährleistet ist. Als Alternative der zufälligen Auswahl der Knoten, wurde auch eine Variante implementiert, bei der es für die Auswahl der Startknoten ebenfalls eine (1xn) Pheromon-Matrix gibt (wobei trotzdem aus jedem Cluster nur ein Knoten gewählt werden kann).

In Abbildung 26 sind zwei verschiedene, durch das oben beschriebene Verfahren generierte, Clustereinteilungen für ein 300-Knoten Problem gezeigt, wobei die grauen Punkte die Kunden zeigen (die Größe ist abhängig vom Bedarf), und die roten Punkte die Cluster-Mittelpunkte. Dieses Bild zeigt deutlich, dass die Aufteilung der Clustermittelpunkte durchaus angemessen erscheint (sie sind über den gesamten Graphen verteilt, wobei bei den Ballungszentren die Dichte der Mittelpunkte größer ist), sich die Clustereinteilungen jedoch deutlich unterscheiden können.

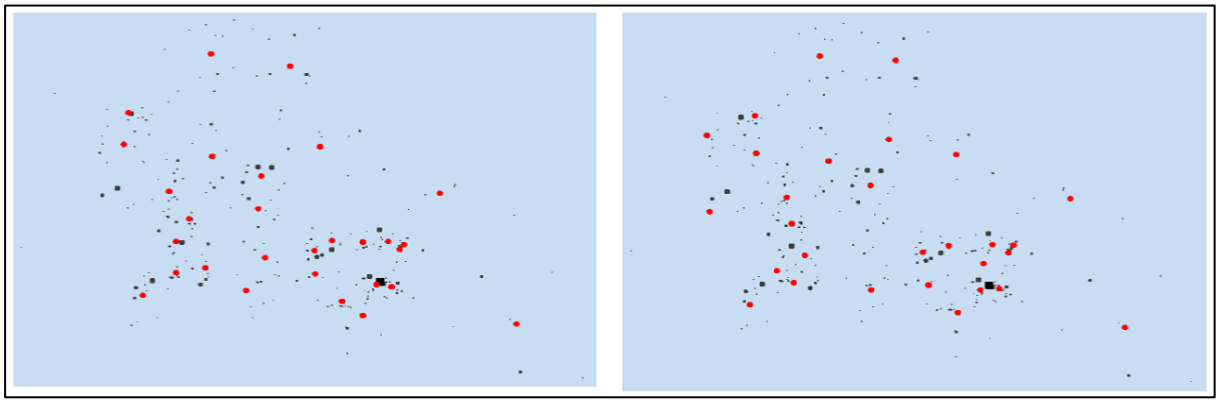


Abbildung 26: Mittelpunkte von verschiedenen Clustereinteilungen für ein 300-Knoten Problem

Da bei dem implementierten Ameisenalgorithmus die zu Grunde liegende Clustereinteilung maßgeblich für den Erfolg der Suche ist, wird erstens jedem Team eine andere Clustereinteilung zugewiesen, und zweitens werden die Cluster von Teams, die keine gute Lösungen finden, regelmäßig durch neue ersetzt. Hier gibt es natürlich auch die Variante, bei der zunächst viele verschiedene Clustereinteilungen ausprobiert werden, und gegen Ende der Suche alle Teams dieselben (besten) Einteilungen verwenden.

Die ersten Tests dieser Variante des Ameisenalgorithmus zeigen schon deutlich bessere Ergebnisse als der im vorigen Abschnitt gezeigte genetische Algorithmus. So konnten bei den zehn getesteten Problemen mit je 300 Knoten die Savings-Ergebnisse um durchschnittlich ca. 5-6% verbessert werden. Die Rechenzeit liegt dabei mit ca. 100 Sekunden je Problem zwar deutlich unter jener des genetischen Algorithmus, jedoch für ein komfortables interaktives Testen verschiedenster Szenarien ist dies auch eher zu lange (man stelle sich vor, fast 2 Minuten vor einem Bildschirm zu sitzen, ohne dass sich etwas Sichtbares tut). Diese Zeit würde sich bei gegebener Problemstellung wahrscheinlich sogar noch deutlich erhöhen, denn bei den Tests wird immer nur ein VRP gelöst, in Wirklichkeit sollen aber bis zu vier gleichzeitig bearbeitet werden. Damit wird sich durch die parallele Verarbeitung der verschiedenen Ameisen-Kolonien, in der Regel zwischen zwei und vier, kein Rechenzeitvorteil mehr ergeben (da ein durchschnittlicher Rechner nicht mehr als vier Prozessoren besitzt).

Doch wenn man von der angestrebten Art der Integration der automatischen Tourenzusammenstellung in das entwickelte Programm absieht, dann kann diesem Verfahren durchaus hohes Potenzial zugeschrieben werden. Denn erstens sind für das Lösen solch komplexer Probleme auch 100 Sekunden sehr wenig (in [33] wird beispielsweise von Rechenzeiten zwischen 30 bis 60 Minuten für das Lösen größerer Problemstellungen berichtet), und zweitens sind die Ergebnisse trotz sicherlich ausbaufähiger Parametrisierung schon sehr gut. So konnte beispielsweise mit der in Abbildung 27 (auf der nächsten Seite) gezeigten Suche das Savings-Ergebnis um über 6% verbessert werden, obwohl die Einstellungen des Verfahrens sicherlich zu „aggressiv" gewählt wurden. Denn wie an dem Verlauf der gleichen Kanten des iterationsbesten Teams in Iteration i und i-1 (rechtes Diagramm) zu sehen ist, stagniert die Suche nach den anfänglichen Neuinitialisierungen der Pheromon-Matrizen sehr stark. An dem flachen Abfall der Kosten des Iterationsbesten Teams (linkes Diagramm) ist zusätzlich zu erkennen, dass auf Grund der Vorgabe der Startknoten der einzelnen Touren durch die Clustereinteilung, bereits zu Beginn der Suche relativ gute Ergebnisse gefunden werden.

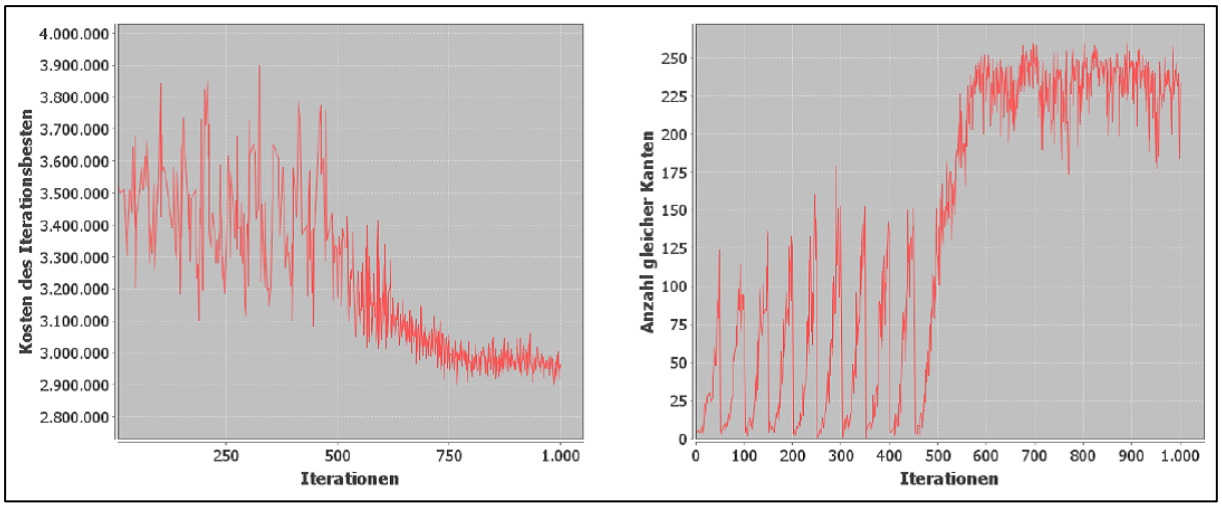


Abbildung 27: Suchverhalten des implementierten Ameisenalgorithmus

Da auch dieses Verfahren für den angestrebten Verwendungszweck zu langsam ist, werden die Auswirkungen der unterschiedlichen Einstellungsmöglichkeiten nicht näher untersucht, sondern im nächsten Abschnitt eine weitere (gänzlich andere) Methode zur schnellen Optimierung erarbeitet.

5.4.3 Der Petal-Algorithmus als Verbesserungsverfahren

Da die zwei bisher gezeigten Verfahren für die in dieser Arbeit behandelten Problemstellung auf Grund der etwas längeren Rechenzeiten leider keine zufriedenstellenden Lösungen bieten, wird an dieser Stelle noch ein weiterer Versuch gestartet, eine zufriedenstellende Optimierung bei einer möglichst kurzen Rechenzeit zu erreichen. Dafür wird nochmals die Idee des beim genetischen Algorithmus bereits verwendeten Petal-Algorithmus aufgegriffen. Wie bereits bei der Beschreibung der implementierten Konstruktionsheurisiken angedeutet, soll dieser jedoch nicht als Konstruktionsverfahren verwendet werden, sondern als (lokales) Verbesserungsverfahren. In Abschnitt 3.3.3 wurde gezeigt, dass mit Hilfe des Petal-Algorithmus zu einer gegebenen Reihenfolge der Knoten die optimale Zusammenfassung zu Touren ermittelt werden kann. Um nun diesen Algorithmus als lokales Verbesserungsverfahren zu verwenden, müssen lediglich die durch eine anderes Verfahren generierten Touren aneinander gereiht, und somit zu einer gesamten Rundreise zusammengeführt, werden (dabei ist natürlich aus jeder Tour das Depot zu entfernen). Dieses Generieren der Giant-Route wird folgendermaßen umgesetzt: Aus der ersten Tour werden die Knoten in der Reihenfolge übernommen, wie sie in dieser Tour enthalten sind. Danach wird aus den übrigen Touren diejenige gesucht, die den, zum letzten Knoten der Giant-Route nächstgelegenen, Knoten besitzt. Von diesem Knoten aus wird anschließend die kürzeste (offene) Route zu den übrigen Knoten dieser Tour generiert (mittels Lin-Kernighan, wobei der ausgewählte Knoten als

Depot fungiert), und an die Giant-Route angehängt. Dieser Vorgang wird solange wiederholt, bis keine Tour, und somit kein Knoten, mehr übrig ist.

Die so ermittelte Giant-Route wird dem Petal-Algorithmus übergeben, dieser erstellt zu jedem Knoten die entsprechenden Petals, bewertet diese, und sucht jenes Petal-Set mit den geringsten Kosten (mittels dem Dijkstra-Algorithmus für das Shortest-Path-Problem). Für die Erstellung der Petals bzw. deren Bewertung gibt es drei verschiedene Parameter, mit denen die Genauigkeit des Algorithmus eingestellt werden kann (wobei eine höhere Genauigkeit natürlich mit einer höheren Laufzeit einhergeht). Mit Hilfe des ersten Parameters kann ein Grenzwert für die minimale Laderaumauslastung des kleinstmöglichen Fahrzeugs angegeben werden. Dies bedeutet, Petals, die Touren mit einer geringeren Laderaumauslastung als diesen Grenzwert repräsentieren, werden nicht berücksichtigt. Der zweite Parameter gibt die Häufigkeit an, mit der die Routen zu einem Petal neu berechnet werden. Dies ist so zu verstehen: Zunächst wird für einen Ausgangsknoten immer das größtmögliche Petal, und somit die größtmögliche Route, konstruiert und bewertet. Anschließend wird solange durch Entfernen des letzten Knotens ein neues Petal erstellt, bis entweder nur noch ein Knoten in der Route ist, oder der oben erwähnte Grenzwert der Laderaumauslastung unterschritten wird. Der Parameter gibt nun an, wie viele Knoten aus der bereits errechneten Route entfernt werden können, ehe eine komplette Neuberechnung der Route notwendig ist. Mit dem letzten Parameter wird angeben, ab welcher Anzahl an Knoten die sukzessiv konstruierte Route mittel dem Lin-Kernighan-Verfahren optimiert wird.

Der Petal-Algorithmus wurde zunächst auf die Ergebnisse der zwei gezeigten Konstruktionsheuristiken (Abschnitt 5.3.1) für die jeweils 100 Probleme mit 200, 300 bzw. 400 Knoten getestet. Diese Tests zeigen, dass die Nearest-Neighbor-Ergebnisse bei 200 Knoten um durchschnittlich 1,7%, bei 300 Knoten um durchschnittliche 1,1% und bei 400 Knoten um durchschnittlich 0,7% verbessert werden können. Bei den Savings-Ergebnissen liegen die Verbesserungsraten bei 1,4, 1,6 bzw. 1,9%. Die durchschnittlich benötigte Rechenzeit liegt zwischen 47ms (für ein 200-Knoten Problem) und 87ms (für ein 400-Knoten Problem). Auffallend bei diesen Ergebnissen ist, dass erstens die bereits deutlich besseren Savings-Ergebnisse noch stärker verbessert werden als die Nearest-Neighbor-Ergebnisse, und zweitens sich die Verbesserungsraten bei den Savings-Lösungen mit steigender Knotenanzahl erhöhen, bei den Nearest-Neighbor-Lösungen jedoch kleiner werden. Dass die Ergebnisse des Nearest-Neighbor-Algorithmus auf diese Weise nicht so stark verbessert werden können, liegt daran, dass durch die langgezogenen Touren, die bei dieser Heuristik entstehen, große Distanzsprünge in der Giant-Route vorhanden sind.

Die Veränderung der durchschnittlichen Laderaumauslastung zeigt auch, dass sich die Ergebnisse der Savings-Lösungen deutlicher ändern. So kann diese mit Hilfe des Petal-Algorithmus bei Savings-Lösungen von ca. 85% auf 90% gesteigert werden – dies entspricht bei 400-Knoten-Problemen einer durchschnittlichen Einsparung von 2 Fahrzeugen. Bei den Nearest-Neighbor-Lösungen ändert sich die Laderaumauslastung hingegen nur minimal: von 96,4% auf 94,7% (bei 200 Knoten) bzw. von 98,2% auf 96,8% (bei 400 Knoten).

Diese Ergebnisse zeigen, dass mit Hilfe dieses Verfahrens Lösungen deutlich verbessert werden können, und dabei vor allem auch der gewünschten Ausgewogenheit zwischen Laderaumauslastung und Gesamtdistanz bzw. Clusterung näher gekommen wird. Weiters werden bei diesem Verfahren auch die unterschiedlichen Fahrzeugtypen berücksichtigt. Da es aber bei der gegebenen Problemstellung lediglich die zwei Varianten mit und ohne Hänger gibt, diese sich aber in den variablen (kilometerabhängigen) Kosten nur geringfügig voneinander unterscheiden, führt dies kaum zu Veränderungen der Lösungen. Lediglich bei den dem Depot sehr nahen Touren kommt es vor, dass die Variante ohne Hänger vorkommt (was sich mit den manuellen Planungen in der Praxis deckt).

Auf Grund der positiven Testergebnisse dieses Verfahrens soll es auch mit dem bisher besten Verfahren, der lokalen Suche mittels λ-Interchanges, gekoppelt werden. Hierbei gibt es neben der Möglichkeit, den Petal-Algorithmus zum Schluss einmalig durchzuführen, auch die Variante, bei der die beiden Verfahren eine Schleife durchlaufen (im weiteren Verlauf der Arbeit als „Petal-Loop" bezeichnet). Das heißt, zunächst wird ein konstruiertes Verfahren mittels λ-Interchanges verbessert, bis durch das Vertauschen einzelner Kunden zwischen zwei Touren keine bessere Lösung mehr gefunden wird. Anschließend wird das Ergebnis dem Petal-Algorithmus übergeben, der auf eben beschriebenen Weise versucht, das Ergebnis weiter zu verbessern. Gelingt dies, wird das Ergebnis wieder der lokalen Suche mittels λ-Interchanges übergeben. Diese Schleife wird so lange durchlaufen, bis keines der beiden Verfahren mehr eine Verbesserung der Lösung bewirkt. Auf Grund der Tatsache, dass die implementierte lokale Suche mittels Knotentausch nicht die unterschiedlichen Fahrzeugtypen berücksichtigt, und der Petal-Algorithmus kleine Ungenauigkeiten bei der Bewertung der Petals aufweist (siehe oben), kann diese Stagnation des Verfahrens leider nicht anhand des Zielfunktionswertes eindeutig erkannt werden. Denn es könnten zum Beispiel dieselben Ergebnisse durch die beiden Algorithmen unterschiedlich bewertet werden, was zu einer Endlosschleife des Verfahrens führen könnte. Deshalb wird die Anzahl der Iterationen auf acht Durchläufe beschränkt. Tests haben gezeigt, dass danach kaum mehr Verbesserungen gefunden werden, und wenn doch, in keinem nennenswerten Ausmaß.

In den Abbildungen 28 und 29 auf den nächsten Seiten sind die Ergebnisse der Testläufe mit verschiedenen Einstellungen gezeigt. Die Abbildung 27 zeigt die Ergebnisse mit den Savings-Startlösungen, wobei für λ=1 und λ=2 (Best-Fit, Nachbarschaftsquote=0,1) jeweils ein Durchgang ohne Petal-Algorithmus, einer mit dem Petal-Algorithmus am Schluss und einer mit der Petal-Algorithmus-Schleife (dem Petal-Loop) durchgeführt wurde – wieder für jeweils 100 Probleme mit 200, 300 bzw. 400 Knoten. Abbildung 29 zeigt die Ergebnisse derselben Testläufe, jedoch mit Nearest-Neighbor-Startlösungen.

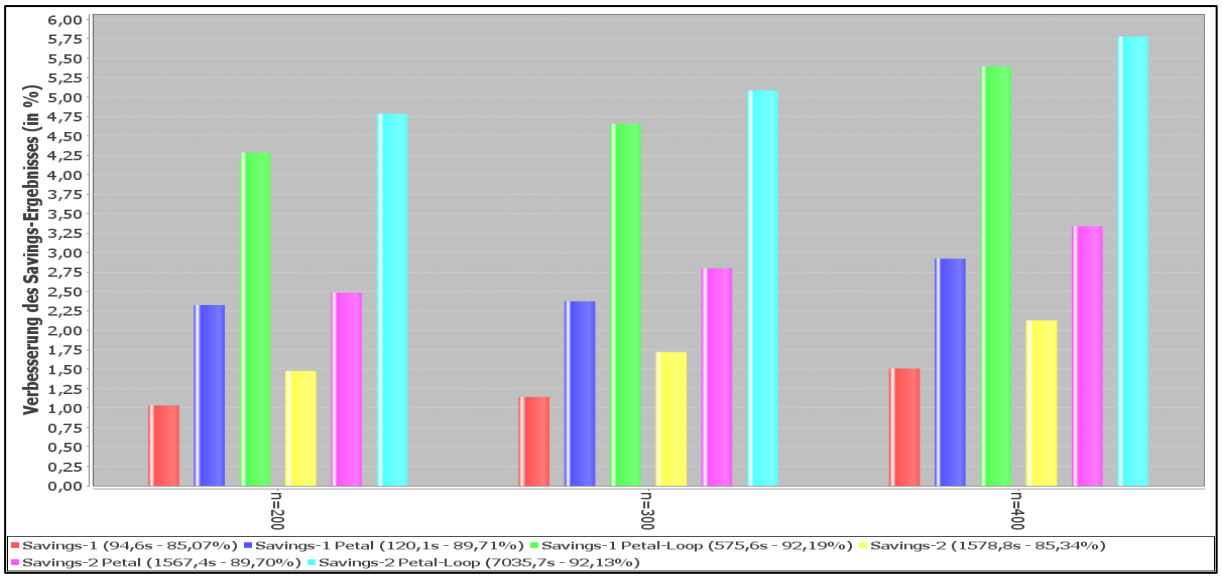


Abbildung 28: lokale Suche in Kombination mit dem Petal-Algorithmus (Savings-Start)

Die Ergebnisse der Testläufe zeigen sehr deutlich, welches Potenzial in dem Petal-Algorithmus als Verbesserungsheuristik steckt: Während mit der herkömmlichen lokalen Suche mit λ=1 die Verbesserungsraten zu den Savings-Ergebnissen bei ca. 1,0-1,5% liegen, liegen die Verbesserungen mit nachgeschalteten Petal-Algorithmus bereits bei etwa 2,3-2,9% und mit dem Petal-Loop bei 4,3 bis 5,4%. Damit sind die erzielten Ergebnisse mit einem λ=1 Petal-Loop deutlich besser als die reine lokale Suche mit λ=2, und das bei knapp einem Drittel der Rechenzeit. So benötigt ein λ=2 Durchgang (mit einer Savings-Startlösung) durchschnittlich 5,3 Sekunden, während bei dem λ=1 Petal-Loop im Schnitt lediglich 1,9 Sekunden vergehen. Wird der Petal-Loop auf die lokale Suche mit λ=2 angewandt, so werden (natürlich) noch bessere Ergebnisse erzielt, wobei die Steigerung der Verbesserungsrate zum Savings-Ergebnis von 4,3-5,4% auf 4,8-5,8% relativ bescheiden ausfällt. Die Erhöhung der Rechenzeit ist jedoch, wie bei der rein lokalen Suche, sehr hoch, denn während mit dem λ=1 Petal-Loop die eben erwähnten 1,9 Sekunden zur Lösung eines durchschnittlichen Problems benötigt werden, wartet man bei dem λ=2 Petal-Loop bereits durchschnittlich 23,5 Sekunden auf das Ergebnis. Deutlich zu sehen ist bei diesen Ergebnissen auch wieder, dass die Verbesserung der Savings-Lösungen insbesondere über die Reduktion der Fahrzeuge (und damit

der Erhöhung der Laderaumauslastung) geht. Denn während bei der einfachen λ=1 Variante die Laderaumauslastung noch bei ca. 85% liegt, wird bei den Ergebnissen des Petal-Loops bereits über 92% des zur Verfügung stehenden Laderaums ausgenutzt.

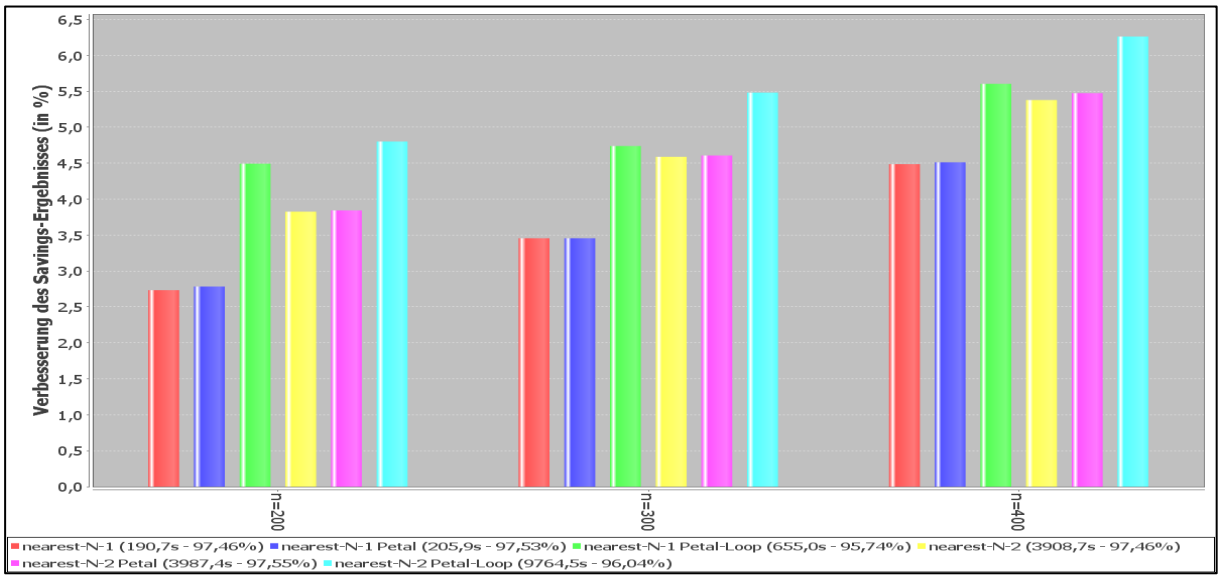


Abbildung 29: lokale Suche in Kombination mit dem Petal-Algorithmus (Nearest-Neighbor-Start)

Wie in Abbildung 29 zu sehen ist, und auch bereits bei den Untersuchungen des Petal-Algorithmus mit den reinen Nearest-Neighbor-Lösungen ersichtlich war, werden die Nearest-Neighbor-Startlösungen auch in Kombination mit der lokalen Suche durch den Petal-Algorithmus nicht so stark verbessert wie die Savings-Lösungen. Bei diesen Startlösungen erzielt die reine lokale Suche mit λ=2 bei den größeren Problemen sogar fast gleich gute Ergebnisse wie der λ=1 Petal-Loop. Insgesamt sind die Ergebnisse mit diesen Startlösungen, wie auch bei der reinen lokalen Suche, besser als mit den Savings-Startlösungen. Der Unterschied der Petal-Loop-Varianten ist jedoch sehr gering: Bei den Problemen mit 200 Knoten gibt es mit 4,78 zu 4,80% nahezu keinen Unterschied der besten Verbesserungsraten (λ=2 Petal-Loop), und auch bei den Problemen mit 400 Knoten ist der Unterschied mit 5,78% zu 6,26% wesentlich geringer als bei der reinen lokalen Suche (2,12% zu 5,48%). Das Verhalten der Rechenzeit zeigt das gewohnte Bild: Die lokale Suche bei Nearest-Neighbor-Startlösungen dauert länger als bei Savings-Startlösungen, und die Erhöhung von λ führt zu einer starken Rechenzeitverlängerung. Die Laderaumauslastung wird mit Hilfe des Petal-Loops zu Gunsten der Entwirrung des Tourenplans verringert, jedoch bleibt die Auslastung bei den Nearest-Neighbor-Startlösungen deutlich über jener der Savings-Startlösungen (96% zu 92%).

In Abbildung 30 auf der nächsten Seite sind die Tourenpläne der Startlösungen, sowie die daraus generierten Lösungen durch den Petal-Loop für ein 300-Knoten-Problem zu sehen. Während die Startlösungen die bereits diskutierten Schwächen aufweisen (zu geringe Lade-

raumauslastung beim Savings-Algorithmus und das Durcheinander der Touren bei dem Nearest-Neighbor-Algorithmus), sind bei den zwei Tourenplänen des Petal-Loops keine Schwächen mehr erkennbar. Auch das im Abschnitt 5.3.2 erwähnte Problem mit der Ausgewogenheit zwischen Laderaumauslastung und Gesamtdistanz bzw. Clusterung kann mit diesem Verfahren gelöst werden. So wird zum Beispiel bei dem gezeigten Problem die Anzahl an Fahrzeugen von 32 auf 30 gesenkt (bei der Savings-Variante), bzw. von 28 auf 29 erhöht (bei der Nearest-Neighbor-Variante). Obwohl sich die Bilder der Tourenpläne dieser beiden Varianten doch meist deutlich voneinander unterscheiden, so liegt die Lösungsqualität in der Regel sehr nahe beieinander (siehe obere Auswertungen). Dies bedeutet mit diesem Verfahren werden je nach Startlösung zwei im Lösungsraum der verwendeten Zielfunktion relativ weit voneinander entfernte lokale Minima gefunden.

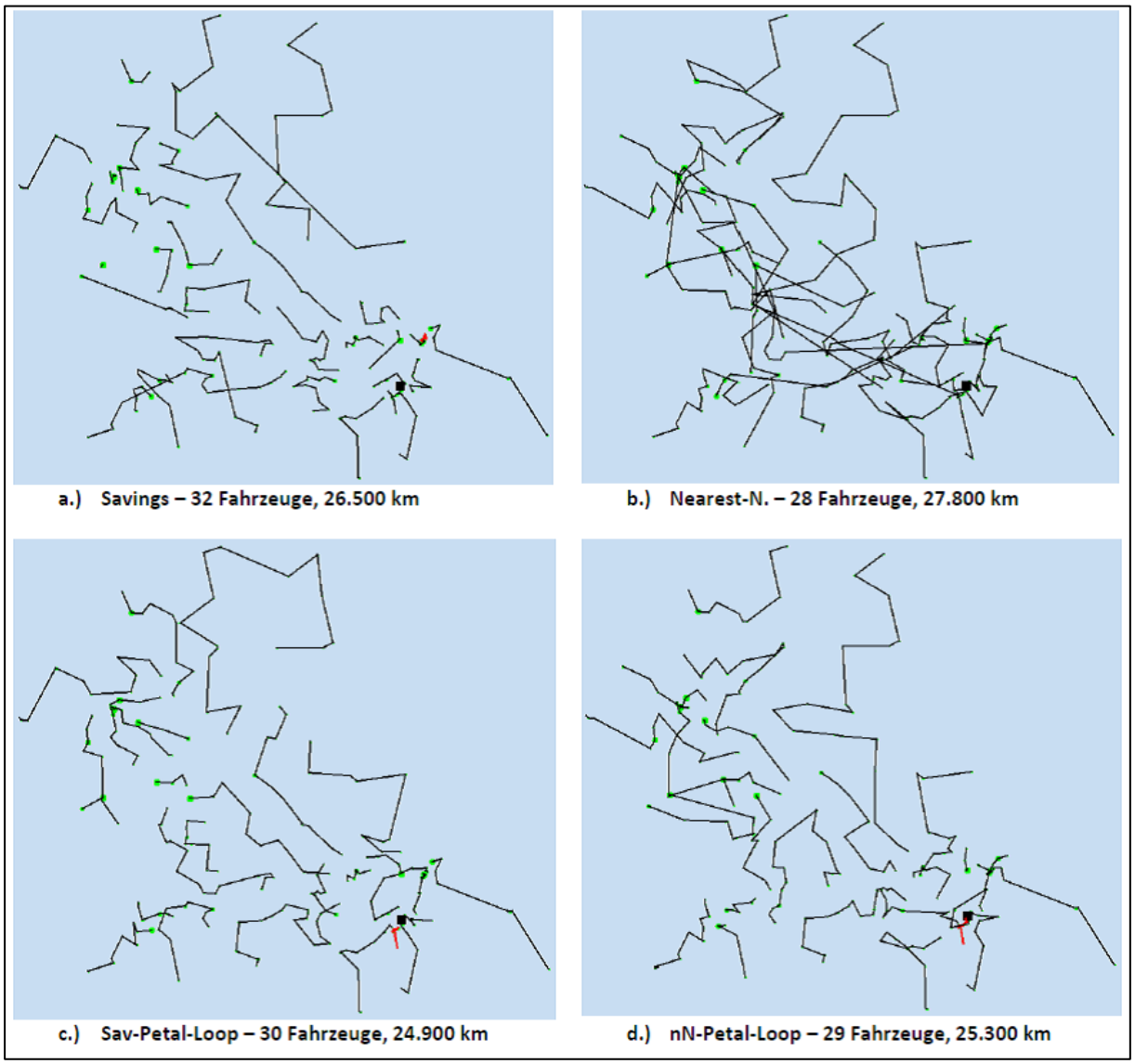


Abbildung 30: Auswirkung des Petal-Loops auf die Tourenpläne der verschiedenen Startlösungen

Anhand der eben gezeigten Auswertungen, und auch anhand der Abbildung der Tourenpläne, ist deutlich zu erkennen, dass das Petal-Loop-Verfahren ein sehr effizientes, und für Problemstellungen dieser Art und Größe extrem schnelles, Verfahren ist. Das Verfahren stellt auf Grund seiner Vorgehensweise zwar „nur" eine lokale Suche dar, und untersucht somit nur einen, durch die Startlösung definierten, Bereich des Lösungsraums, doch scheinen die dadurch erzielten Ergebnisse für den angestrebten Einsatz durchaus ausreichend.

Denn erstens kann davon ausgegangen werden, dass der Zielfunktionswert der mit Hilfe des Petal-Loop-Verfahrens generierten Lösungen nicht wesentlich höher als jener des globalen Minimums ist. Diese Annahme kommt nicht nur daher, dass weder durch Vergrößerung der Nachbarschaft, noch durch den im vorigen Abschnitt gezeigten Ameisenalgorithmus wesentlich höhere Verbesserungsraten erzielt werden können, sondern stützt sich auch auf die publizierten Ergebnissen in [6]. Dort wird berichtet, dass mit dem einfachen Savings-Verfahren (in der angeführten Literatur als Clark and Wright-Verfahren bezeichnet) Ergebnisse erzielt werden, die durchschnittlich nur 6,7% von der besten Lösung entfernt sind. Wie gezeigt wurde, wird mit dem Petal-Loop-Verfahren das Savings-Ergebnis um ca. 4,8 bis 5,8% verbessert, wobei der hier verwendete Savings-Algorithmus bereits eine sukzessive Konstruktion der Routen bei der Zusammenlegung zweier Touren verwendet, sowie eine ständige Neuberechnung der Savings und eine Optimierung der Routen mittels dem Lin-Kernighan-Verfahren integriert (also an sich schon eine Verbesserung des Savings-Algorithmus ist). Zweitens wird die Zielerreichung ohnehin nur auf Basis einer theoretischen Kostenfunktion, die den Gegebenheiten der tatsächlich vorliegenden Problemstellung zwar angepasst wird, diese jedoch sicherlich nicht exakt widergibt, gemessen. Dies bedeutet, auch wenn es Lösungen mit einem geringeren Zielfunktionswert gibt, so kann trotzdem nicht sicher gesagt werden, ob diese Lösung auch tatsächlich besser ist. Drittens werden bei der untersuchten Problemstellung überwiegend Frächter beauftragt, die für jede Tour einen Pauschalpreis verrechnet, weshalb eine geringfügige Änderung der Gesamtdistanz (und somit des Zielfunktionswertes) nicht unmittelbar zu einer tatsächlichen Reduktion der Transportkosten führen muss. Als letzter wichtiger Grund sei nochmals an die angestrebte Art der Integration dieser automatischen Tourenzusammenstellung in den gesamten Tourenplanungsprozess erinnert: Dem Tourenplaner soll es ermöglicht werden, mehrere Testläufe zu machen, dabei verschiedene Szenarien zu testen, und somit auf das erzielte Ergebnis Einfluss zu nehmen. Für diesen Zweck ist dieses Verfahren sehr gut geeignet, denn erstens ist es sehr schnell, und zweitens können dem Benutzer des Programms zwei Varianten zur Verfügung gestellt werden, die nahezu gleichwertige Ergebnisse erzielen, aber bezüglich der Tourenzusammenstellung sehr unterschiedlich sind. So kann der Tourenplaner zum Beispiel von einer

Variante ein paar Touren übernehmen und die übrigen Aufträge mit der zweiten Variante verplanen. Ein weiterer Vorteil dieses Verfahrens ist es, dass durch λ die Intensität der Suche eingestellt werden kann. So kann für das schnelle Testen von grundlegend verschiedenen Szenarien die Variante mit λ=1 verwendet werden, und für das genauere Untersuchen der ausgewählten Alternativen die intensivere Suche mit λ=2.

6 Conclusio

Im Zuge dieser Arbeit wurde für das Unternehmen Elefant Holding AG ein neues Konzept für die konzernweite Tourenplanung entworfen, und dafür das Programm inklusive automatischer Tourenzusammenstellung erstellt. Nach erfolgter Umsetzung dieses Konzepts (und Implementierung des Programms) gilt es nun, ein Resümee über die Ergebnisse dieser Arbeit bzw. den erzielten Nutzen zu ziehen. Dabei werden die positiven Effekte auf die drei Bereiche Reduktion des administrativen Aufwands, Reduktion der Transportkosten, sowie positive Nebeneffekte aufgeteilt.

Die verschiedenen Ansatzpunkte für die **Reduktion des administrativen Aufwands** können am besten durch Vergleich der wesentlichen Prozessschritte innerhalb der Auftragsbearbeitung (vor und nach der Umsetzung) gezeigt werden: Bisher kam jeder Kundenauftrag zur Vorschreibung der geplanten Auslieferungswoche zum Tourenplaner des jeweiligen Werkes, ehe dieser erfasst werden konnte. Nach der Erfassung des Auftrags kam der Auftrag in die Auftragskontrolle, erst danach kam der Auftrag (in Papierform) wieder zum Tourenplaner. Seit der Umstellung kann der Auftrag sofort erfasst werden, da es innerhalb der Auftragserfassung möglich ist, für diesen die möglichen Lieferwochen abzufragen. Dies wird dadurch ermöglicht, dass das gesamte Absatzgebiet in Liefergebiete eingeteilt ist, und diesen bestimmte Lieferrhythmen zugeteilt sind. Über das Länderkennzeichen und die Postleitzahl wird dem Auftrag das entsprechende Liefergebiet und damit die möglichen Lieferwochen, zugeordnet. Der Tourenplaner kann sich eine Übersicht der bereits zugesagten Aufträge (aufgeteilt auf die Liefergebiete) anzeigen lassen, und gegebenenfalls Wochen „sperren" oder „öffnen". Unmittelbar nach der Erfassung des Auftrags ist dieser innerhalb des Tourenplanungsprogramms ersichtlich, und kann somit sofort bearbeitet werden – der Auftrag wird innerhalb der Tourenplanung nicht mehr in Papierform benötigt.

Eine wesentliche Aufwandsreduktion konnte durch das entwickelte Programm natürlich bei der Kernaufgabe der Tourenplanung, der Tourenzusammenstellung, erzielt werden. Bisher sortierte der Disponent alle Aufträge anhand des Liefergebietes in verschiedene Ablagefächer ein, und führte Listen mit den geschätzten Volumina. Zur tatsächlichen Tourenzusammenstellung wurden diese Aufträge wieder aus den Ablagen genommen und zu Touren zusammengesteckt, wobei eine Liste mit den Abladestellen, sowie den jeweiligen Volumina, manuell geführt werden musste. Die Routen der Touren wurden mit Hilfe einer großen Landkarte erstellt. Wenn Kunden gemeinsam beliefert werden sollten, so mussten sich die Tourenpla-

ner der jeweiligen Werke (per Telefon) gut absprechen, denn beide hatten nur Zugang zu den standorteigenen Aufträgen. Mit Hilfe des entwickelten Programms kann der Benutzer nun alle Aufträge aller Standorte sehen, und diese unter anderem nach Standort, Liefergebiet und Status selektieren, oder auf die einzelnen Lieferadressen verdichten. Dabei werden ihm immer die wichtigsten Informationen (vor allem die bestellten Volumina) angezeigt und die Aufträge auf einer Karte aufbereitet. Bei einer manuellen Tourenzusammenstellung wird der Tourenplaner durch zahlreiche Funktionen unterstützt. So kann er zum Beispiel mit gedrückter Maustaste über die Karte fahren, und ihm wird für das dadurch gezeichnete Rechteck immer angezeigt, wie viele Kunden sich gerade in dem markierten Gebiet befinden, und welchen Gesamtbedarf diese haben. Für die erstellten Touren wird ihm neben der prozentuellen (farblich aufbereiteten) Laderaumauslastung auch die Gesamtdistanz in Kilometer oder die errechnete Dauer angezeigt. Die Route wird mittels sukzessiver Konstruktion erstellt, und kann mit einem Mausklick mittels Lin-Kernighan-Verfahren optimiert werden. Zusätzlich gibt es natürlich die in dieser Arbeit erarbeitete Lösung zur automatischen Tourenzusammenstellung, die innerhalb weniger Sekunden eine komplette Einteilung der Aufträge liefert.

Bei der Übertragung der Tourenzusammenstellung in das ERP-System konnte ebenfalls eine sehr hohe Aufwandsreduktion erzielt werden: Während bisher die Tourenzuteilung für jeden Auftrag manuell eingeben werden musste, erfolgt dies nun automatisch. Darüber hinaus wurden durch die zentrale Informationsspeicherung zahlreiche kleinere administrative Prozesse erleichtert. So muss jetzt zum Beispiel bei Auftragsänderungen nicht mehr der Papierauftrag gesucht und ausgebessert werden und auch Lieferauskünfte werden auf Grund der zentralen Speicherung der lieferrelevanten Informationen wesentlich schneller abgewickelt.

Durch die hier angeführten Veränderungen kann nun die Aufgabe der gesamten Tourenplanung, die bisher von einer zentralen Leitung und vier Disponenten betreut wurde, von drei Personen (der zentralen Leitung plus zwei Disponenten) bewerkstelligt werden. Die Tourenbetreuung, mit der bisher konzernweit etwa zwölf Mitarbeiter betraut waren, erfolgt nun durch insgesamt acht Mitarbeiter. Dies alleine entspricht bereits einer Reduktion der Personalkosten von etwa 85.000 € pro Jahr, obwohl der meiste Aufwand in den Oststandorten (mit geringeren Lohnkosten) wegfällt.

Eine noch viel größere Einsparung wird durch die effizientere Tourenplanung, und damit durch die **Reduktion der Transportkosten**, erzielt. Bereits durch den Einsatz des Programms für eine manuelle Einteilung der Aufträge kann auf Grund der übersichtlichen Informationsaufbereitung von einer effizienteren, und damit kostengünstigeren, Tourenzusammenstellung (gegenüber der bisherigen Vorgehensweise) ausgegangen werden. Für eine

seriöse Bewertung dieses Effekts ist das Programm jedoch leider erstens zu kurz im Einsatz, und zweitens wäre es nur mit sehr großem Aufwand möglich, die Tourenzusammenstellungen der Vergangenheit zu rekonstruieren (und somit einen Vergleichswert herzustellen). Doch alleine die Verbesserung des mit Hilfe des Programms manuell zusammengestellten Tourenplans durch die automatische Tourenzusammenstellung ist beachtlich. Für den Vergleich wurde eine vergangene Woche ausgewählt, und die manuelle Zusammenstellung (so wie die Aufträge tatsächlich ausgeliefert wurden) einer automatisch generierten gegenübergestellt. Das Ergebnis ist in Abbildung 31 zu sehen. In der ausgewählten Kalenderwoche wurden von dem österreichischen Standort insgesamt 322 Kunden beliefert, wobei der Gesamtbedarf ca. 2.300 m³ ausmachte. Die Auslieferung erfolgte (auf Basis der manuellen Zusammenstellung) durch 29 Fahrzeuge, die eine Gesamtdistanz von knapp 23.000 km zurücklegten. Die automatische Tourenzusammenstellung (mittels einem λ=1 Petal-Loop-Verfahren mit Savings-Start) zeigt nach nur ca. drei Sekunden (inklusive grafischer Aufbereitung) einen Tourenplan, bei dem nur 25 Fahrzeuge mit einer Gesamtdistanz von etwa 20.000 km benötigt werden. Dies entspricht sowohl für die Anzahl der Fahrzeuge als auch für die Gesamtdistanz eine Einsparung von ca. 15%.

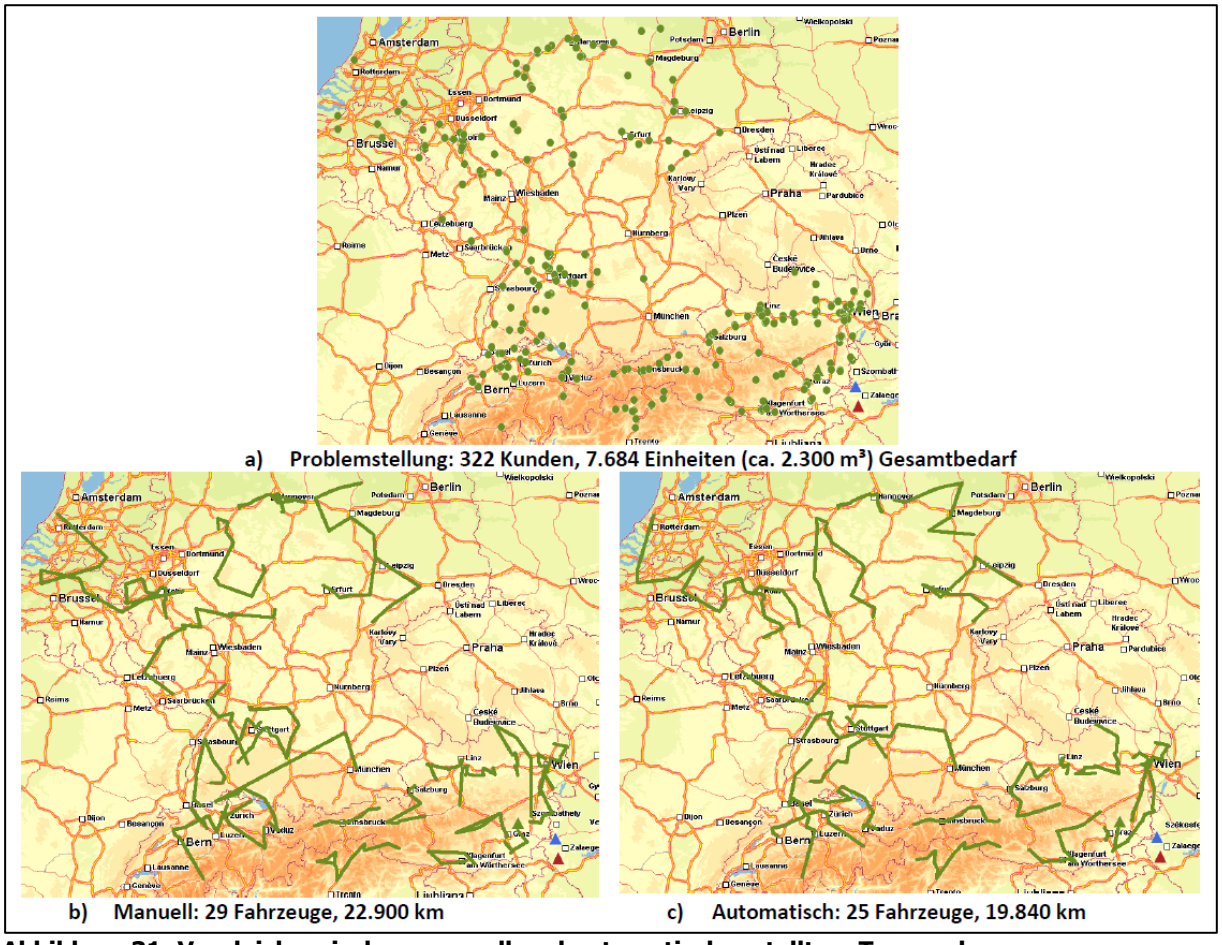


Abbildung 31: Vergleich zwischen manuell und automatisch erstelltem Tourenplan

Da sowohl die Anzahl der Fahrzeuge, als auch die Gesamtdistanz um etwa denselben Faktor verringert werden konnten (bei einer mindestens gleichbleibenden Clusterung), und diese sicherlich die zwei wesentlichsten Kostentreiber darstellen, kann davon ausgegangen werden, dass die Reduktion der Transportkosten nicht wesentlich geringer ausfallen wird. Bei jährlichen Transportkosten von ca. 7,7 Mio. € entspricht eine 15%ige Reduktion einer Einsparung von über 1,1 Mio. € pro Jahr.

Die eben gezeigte Reduktion der Transportkosten sowie die Reduktion des administrativen Aufwands stellen sicher die zwei wesentlichen Vorteile der durchgeführten Neustrukturierung bzw. der Implementierung des Programms dar. Dass durch diese Änderung aber auch zahlreiche andere **positive Nebeneffekte** erzielt werden konnten, soll nicht ganz unerwähnt bleiben, weshalb an dieser Stelle ein paar Beispiele für diese gezeigt werden. Als erster solcher Nebeneffekt ist die verkürzte (administrative) Durchlaufzeit zu erwähnen, die mit der Aufwandsreduktion einhergeht. Die Verkürzung macht im Schnitt ein bis drei Tage aus, was insbesondere für Matratzenbestellungen (bei denen eine kürzere Lieferzeit einzuhalten ist) von großem Vorteil ist. Ein weiterer wichtiger Nebeneffekt ist die bereits erwähnte einfachere Berücksichtigung der Produktionsauslastungen. Durch die Anzeige der Planminuten für die drei wichtigsten Produktionsbereiche je Fahrzeug, und in weiterer Folge je Tag, kann der Grundstein für eine ausgeglichene Produktionsauslastung bereits in der Tourenplanung gelegt werden. Als letzter dieser positiven Nebeneffekte sei noch die erhöhte Termintreue erwähnt, die dadurch zu Stande kommt, dass durch die bessere Übersicht über die Auftragslage in den verschiedenen Gebieten, eine Änderung der Lieferwoche nicht mehr so häufig notwendig ist. Mit dem größten Kunden des Unternehmens wird diesbezüglich nun auch noch enger zusammengearbeitet. Diesem werden regelmäßig die Daten zu den nächstmöglichen Lieferterminen in die verschiedenen Gebiete übermittelt, und diese werden in das ERP-System des Kunden übernommen. Damit kann in jedem Möbelhaus des Kunden per Knopfdruck die nächstmögliche Lieferwoche abgefragt und dem Endkunden somit eine genauere Auskunft gegeben werden.

Daraus ist ersichtlich, dass durch die Kombination der Neustrukturierung des Tourenplanungsprozesses und der Einführung des entwickelten Programms mit der Möglichkeit der automatischen Tourenzusammenstellung, in dem Unternehmen erhebliche Vorteile und vor allem Einsparungen, erzielt werden konnten. Hier hat sich die Entscheidung zu einer individuellen neuentwickelten Lösung (gegenüber fertigen Produkten) sicherlich als richtig herausgestellt, denn nur so fanden die unternehmensspezifischen Gegebenheiten und auch Anforderungen bestmögliche Berücksichtigung. So wurde das Programm (und auch die automati-

sche Tourenzusammenstellung) an den neuen Tourenplanungsprozess angepasst, und nicht umgekehrt. Weiters kann solch eine Eigenentwicklung besser (und in der Regel auch kostengünstiger) weiterentwickelt und den sich laufend verändernden Anforderungen angepasst, werden.

Literaturverzeichnis

[1] Applegate, D.; Cook, W.; Rohe, A.: Chained Lin-Kernighan for Large Traveling Salesman Problems. INFORMS Journal on Computing 15, 82-92 (2003)

[2] Beasley, J.E.: Route first-cluster second methods for vehicle routing. Omega The International Journal of Management Science 11, S.403-408 (1983)

[3] Chambers, L.: Practical Handbook of Genetic Algorithms. CRC Press, Florida (1995)

[4] Christofides N.; Mingozzi, A.; Toth, P.: The vehicle routing problem. in: Christofides N.; Mingozzi, A.; Toth, P.; Sani, C.: Combinatorial Optimization. Wiley, Chichester/ New York (1979)

[5] Clarke, G; Wright, J.W.: Scheduling of vehicles from a central depot to a number of delivery points. Operations Research 12, 568-581 (1964)

[6] Cordeau, J.-F.; Gendreau, M.; Laporte, G.; Potvin, J.-Y.; Semet F.: A guide to vehicle routing heuristics. Journal of the Operations Sesearch Society 53, 512-522 (2002)

[7] Cordeau, J.F.; Laporte, G.: Tabu Search Heuristics for the Vehicle Routing Problem. GERAD, Montreal (Forschungsbericht) (2002)

[8] Dantzig, G.B.;Ramser, J.H.: The Truck Dispatching Problem. Management Science 6, 80-91 (1959)

[9] Domschke, W.; Scholl, A.: Logistik: Rundreisen und Touren. Springer, München (2010)

[10] Dorigo, M.; Gambardella, L.M.: Ant Colony System: A Cooperative Learning Approach to the Travelling Salesman Problem. IEEE Transactions on Evolutionary Computation 1, S.53-66 (1997)

[11] Dorigo, M.; Maniezzo, V.; Colorni, A.: Ant System: Optimization by a Colony of Cooerating Agents. IEEE Transactions on Systems, Man, and Cybernetics Part B, 26, S.29-41 (1996)

[12] Dorigo, M.; Stützle, T.: Ant Colony Optimization. MIT Press, Cambridge/ Massachusetts (2004)

[13] Foster, A.; Ryan, D.M.: An integer programming approach to the vehicle scheduling problem. Operations Research 27, S.367-384 (1976)

[14] Gendreau, M.; Hertz, A; Laporte, G.: A Tabu Search Heuristic for the Vehicle Routing

Problem. Managament Science 40, S.1276-1290 (1994)

[15] Gendreau, M.; Laporte, G.; Potvin, J.Y.: Metaheuristics for the Capacitated VRP. In Toth, P.; Vigo, D.: The Vehicle Routing Problem. SIAM, Philadelphia (2002)

[16] Gendreau, M.; Potvin, J.Y., Bräysy, O.; Hasle, G.; Lokketangen, A.: Metaheuristics for the Vehicle Routing Problem and Its Extensions: A Categorized Bibliography. In Golden, B.; Raghavan S.; Wasil, E.: The Vehicle Routing Problem: Latest Advances and New Challenges. Springer, New York (2008)

[17] Gendreau, M.; Tarantilis, C.D.: Solving large-scale vehicle routing problems with time windows: The state-of-the-art. Working Paper, CIRRELT 4 (2010)

[18] Gillet, B.E.; Miller, L.R.: A heuristic algorithm for the vehicle-dispatch problem. Operations Research 22, S.340-349 (1974)

[19] Glover F.: Future Paths for Integer Programming and Links to Artificial Intelligence. Computers and Operations Research 13, S.533-549 (1986)

[20] Golden, B.; Raghavan S.; Wasil, E.: The Vehicle Routing Problem: Latest Advances and New Challenges. Springer, New York (2008)

[21] Golden, B.; Wasil, E.; Kelly, J.; Chao, I-M.: The impact of metaheuristics on solving the vehicle routing problem: Algorithms, problem sets, and computional results. in Crainic, T.; Laporte, G.: Fleet Management and Logistics. Springer, Boston (1998)

[22] Helsgaun, K.: An effective implementation of the Lin-Kernighan traveling salesman heuristic. European Journal of Operational Research 126, 106-130 (2000)

[23] Holland, J.H.: Adaptation in Natural and Artificial Systems. MIT Press, Ann Arbor (1975)

[24] Homberger, J.; Gehring, H.: Two Evolutionary Meta-heuristics for the Vehicle Routing Problem with Time Windows. INFOR 37, S.297-318 (1999)

[25] Kindervater, G.A.P.; Savelsbergh, M.W.P.: Vehicle routing: handling edge exchanges. in Aarts, E.H.L.; Lenstra, J.K.: Local Search in Combinatorial Optimization. Wiley, Chichester/ New York (2003)

[26] Korte, B.; Vygen, J.: Kombinatorische Optimierung. Springer, Berlin Heidelberg (2008)

[27] Laporte, G.; Semet, F.: Classical Heuristics for the Capacitated VRP. In Toth, P.; Vigo, D.: The Vehicle Routing Problem. SIAM, Philadelphia (2002)

[28] Lin, S.; Kernighan, B. W.: An Effective Heuristic Algorithm fort the Traveling-Salesman Problem. Operations Research Vol. 21/2, 498-516 (1973)

[29] Michalewicz, Z.: Genetic Algorithms + Data Structures = Evolution Programs. Springer, Berlin Heidelberg (1999)

[30] Osman, I.H.: Metastrategy Simulated Annealing and Tabu Search Algorithms for the Vehicle Routing Problem. Annals of Operations Research 41, S.421-451 (1993)

[31] Pereira, F.B.; Tavares, J.: Bio-inspired Algorithms for the Vehicle Routing Problem. Springer, Berlin Heidelberg (2009)

[32] Potvin, J.Y.: A Review of Bio-inspired Algorithms for the Vehicle Routing. In Pereira, F.B.; Tavares, J.: Bio-inspired Algorithms for the Vehicle Routing Problem. Springer, Berlin Heidelberg (2009)

[33] Prins, C.: A simple and effective evolutionary algorithm for the vehicle routing problem. Computer & Operations Research 31, S.1985-2002 (2004)

[34] Reimann, M.; Doerner, K.; Hartl, R.F.: D-Ants: Savings based ant divide and conquer the vehicle routing problem. Computer & Operations Research 31, S.563-591 (2004)

[35] Reimann, M; Stummer, M; Doerner, K.: A Savings based Ant System for the Vehicle Routing Problem. in Langdon, W.B.: Proceedings oft he Genetic and Evolutionary Computation Conference (GECCO 2002), San Francisco, S.1317-1325 (2002)

[36] Renauld, J.; Boctor, F.F.; Laporte, G.: An improved petal heuristic for the vehicle routing problem. Journal oft he Operational Research Society 47, S.329-336 (1996)

[37] Rochat, Y.; Taillard, E.D.: Probabilistic Diversification and Intensification in Local Search for Vehicle Routing. Journal of Heuristics 1, S.147-167 (1995)

[38] Ryan, D.M.; Hjorring, C.; Glover, F.: Extensions of the petal method for the vehicle routing. Journal oft he Operational Research Society 44, S.289-296 (1993)

[39] Sadouni, K.: Heterogeneous Fleet Vehicle Routing Problem with Time Windows and Nonlinearly Penalized Delays. Journal of Applied Sciences 6 (9), 1969-1973 (2006)

[40] Taillard, E.D.: Parallel Iterative Search Methods for Vehicle Routing Problem. Networks 23, S.661-673 (1993)

[41] Toth, P.; Vigo, D.: An Overview of Vehicle Routing Problems. In Toth, P.; Vigo, D.: The Vehicle Routing Problem. SIAM, Philadelphia (2002)

[42] Toth, P.; Vigo, D.: The Vehicle Routing Problem. SIAM, Philadelphia (2002)

[43] Toth, P.; Vigo, D.: The granular tabu search and its application to the vehicle routing problem. INFORMS Journal on Computing 15, S.333-346 (2003)

[44] Wenger, W.: Multikriterielle Tourenplanung. GABLER, Wiesbaden (2010)