1 Zwei auf einen Streich: Optimierte dynamische Einsatzplanung für Gelbe Engel und Lastenaufzüge

Jörg Rambau und Cornelius Schwarz
Lehrstuhl Wirtschaftsmathematik
Fakultät für Mathematik, Physik und Informatik
Universität Bayreuth
95440 Bayreuth
E-Mail: {joerg.rambau,cornelius.schwarz}@uni-bayreuth.de

Keywords: online optimization, real-time optimization, reoptimization, integer programming, elevator group control, fleet assignment, vehicle routing
MSC: **90B06**, 90B90, **90C10**, 90C27, 90C35, 90C90

Wir modellieren zwei verschiedene dynamische Einsatzplanungsprobleme: die dynamische Einsatzplanung Gelber Engel beim ADAC und die Steuerung von Lastenaufzügen in einem Versandlager der Herlitz PBS AG. Wir benutzen eine Reoptimierungspolitik, die die Steuerung des Systems mit Hilfe der Lösung von statischen Schnappschussproblemen durchführt. Für die auftretenden Schnappschussprobleme vergleichen wir zwei Modellierungsansätze (Flussmodell versus Tourenmodell), von denen nur einer echtzeittauglich ist. Das Verfahren zur dynamischen Einsatzplanung Gelber Engel ist beim ADAC in Betrieb.

1.1 Aufzüge und Gelbe Engel

Jeder kennt das: Warten auf den Aufzug, aber der Aufzug kommt nicht. Schlimmer noch: Er scheint andere Etagen zu bevorzugen, immer die falsche Richtung zu wählen usw. usf. Die Menschen sind schon vor Jahrzehnten zum Mond geflogen, aber so etwas „Einfaches“, wie einen Aufzug zufriedenstellend zu steuern, können sie offenbar nicht.

Auch wir haben zu diesem speziellen Problem keine endgültige Lösung. Insbesondere da die meisten Aufzugsysteme der Steuerung recht wenig Informationen übermitteln: wie viele Leute in welche Etage fahren wollen, wird nur bei sehr modernen Systemen über eine Zielrufsteuerung erhoben.

Wir stießen auf ein etwas zugänglicheres Aufzugsteuerungsproblem in einem Versandlager der Firma Herlitz PBS AG in Falkensee, nahe Berlin. In diesem Lager gibt es zwei Türme mit je fünf Vertikalfördergeräten für jeweils eine Euro-Palette. Locker formuliert besteht die Aufgabe darin, durch geeignete Steuerung den reibungslosen Betrieb auch unter Hochlast zu gewährleisten.

Die Paletten können auf einem Rundkurs aus Rollen- und Kettenförderbändern die fünf Aufzüge umkreisen, bis sie in einem freien Platz vor einem Aufzug auf Beförderung warten. Der Steuerung sind die Paletten bekannt, sobald sie in den Rundkurs einbiegen (Genaueres in [KR02]).

Dieses Förderlayout ist wohl begründet, für unsere Zwecke in diesem Artikel aber zu kompliziert. Stellen wir uns einfach vor, Paletten warten in jeder Etage in einer „globalen“ Warteschlange auf Beförderung. Sobald sie von der Steuerung einem Aufzug zugewiesen wurden, werden sie in eine „lokale“ Warteschlange vor dem jeweiligen Aufzug befördert. Nun kann der Aufzug nicht mehr gewechselt werden. Im Unterschied zum Originalsystem warten die Paletten nun anstatt zu kreisen (kreisen ist beim Management immer beliebter als warten, weil sich etwas bewegt) und die Zeit, die ein Auftrag benötigt, um von der globalen Warteschlange in eine lokale überführt zu werden, ist für alle Aufzüge gleich. Ferner nehmen wir an, dass die lokalen Warteschlangen unendliche Kapazität haben, während im Originalsystem vor jedem Aufzug nur genau eine Palette Platz hat. Wir möchten ohne Präemption arbeiten. Im Produktionsumfeld bedeutet dabei *Präemption*, dass die Bearbeitung eines Auftrages unterbrochen werden kann, um später fortgesetzt zu werden. Beim Aufzugssteuerungsproblem würde dies bedeuteten, dass eine aufgenommene Paletten in einer Zwischenetage abgesetzt wird und erst später in ihre Zieletage geliefert wird. In der Zwischenzeit könnte der Aufzug dann andere Paletten transportieren. Dies wollen wir ausschließen.

Die Aufgabe, dieses idealisierte Aufzugsystem zu steuern, lautet wie folgt: Weise jeder Palette, die mit gegebener Zieletage in der globalen Warteschlange ankommt, einen Aufzug zu und spezifiziere für jeden frei werdenden Aufzug die nächste zu befördernde Palette, so dass alles unter Hochlast „reibungslos“ läuft.

Szenenwechsel: Pech gehabt! Eine Panne auf der Autobahn! Zum Glück gibt es den ADAC. Etwa 30 Minuten nach dem Anruf aus der Notrufsäule steht ein Hilfefahrzeug beim Havaristen. Damit das funktioniert, muss zwischen dem Anruf in einer ADAC-Hilfezentrale und der Ankunft eines gelben Engels eine Menge organisiert werden.

Zum Beispiel: Welches von etwa 80 gerade verfügbaren Fahrzeugen (Einheiten) soll den Auftrag – einen von etwa 200 – erledigen? In welcher Reihenfolge soll ein Hilfefahrzeug die ihm zugewiesenen Aufträge abarbeiten? Und wie kann man Vertragspartner kostenschonend einsetzen? Wie soll man damit umgehen, dass man zukünftige Aufträge nicht kennt? Auch hier soll ohne Präemption gearbeitet werden, d. h. der Service an einem Havaristen wird nicht unterbrochen (Genaueres in [GKRT02]).

Der ADAC hat mit Hilfe von mathematischen Modellen und Algorithmen den vormals händisch geplanten Vorgang nun automatisiert unterstützt – dazu musste jede der genannten Fragen beantwortet werden.

Die Aufgabe, Gelbe Engel in Echtzeit zu steuern, lautet auf den Punkt gebracht: Bestimme für jede frei werdende Einheit den nächsten zu bearbeitenden Auftrag aus allen bekannten unbearbeiteten Aufträgen – falls vorhanden –, so dass auch unter Hochlast alles „reibungslos“ läuft.

Was haben nun Aufzüge mit Gelben Engeln zu tun? Wir werden sehen.

1.2 Modellierung dynamischer Steuerungsprobleme

Starten wir mit den Aufzügen. Es hat sich bewährt, zielorientiert zu modellieren. Der reibungslose Ablauf muss konkretisiert werden.

Modellierungsprozess 1.2.1
Kläre die Fragen: Was sind die relevanten beobachtbaren Systemzustände? Was sind die relevanten freien Steuerentscheidungen? Was sind die Nebenbedingungen? Was ist das Gesamtziel? Lässt es sich durch eine Kennzahl (Kostenfunktion auf Zuständen und Entscheidungen) charakterisieren?

Das Aufzugsystem ist gekennzeichnet durch die Position der Aufzüge, die gerade beförderten Aufträge und die unbearbeiteten Aufträge. Da wir ohne Präemption arbeiten, können wir annehmen, dass die Aufträge in Bearbeitung direkt zu ihrem Ziel gebracht werden und danach der Aufzug wieder (für neue Aufträge) verfügbar ist. Also sind die relevanten Informationen: für jeden Aufzug seine Daten sowie der Zeitpunkt und die Position der frühesten Verfügbarkeit; ferner die Daten der noch unbearbeiteten Aufträge (Zustände). Entscheiden können wir für unbearbeitete und noch nicht zugewiesene Aufträge die Zuweisung auf einen Aufzug und für jeden Aufzug den Auftrag, der nach dem Freiwerden bedient werden soll (Entscheidungen). Zu beachten ist, dass alle Aufträge in endlicher Zeit bedient werden (Nebenbedingungen). In unserem Aufzugsystem bedeutet reibungsloser Ablauf, dass das Förderband nicht überläuft und dass die Paletten nicht so lange warten müssen. Bleiben wir für den Moment beim zweiten Punkt. Wir wollen so steuern, dass die durchschnittliche Wartezeit der Paletten am Ende des Tages minimal ist und dass die individuellen Wartezeiten wenigstens endlich sind. Das kann durch eine Funktion der Wartezeiten geschehen, die streng konvex ist, da dann die Grenzwartekosten echt steigen und individuell große Wartezeiten überproportional bestraft werden. Ein Beispiel ist die Minimierung einer passenden Skalierung der euklidischen Norm der Wartezeiten (Kosten).

Um praxistaugliche Modelle für die Steuerung komplexer Abläufe zu entwickeln, muss man sich nun über die Verfügbarkeit von Informationen im Klaren sein.

Modellierungsprozess 1.2.2
Kläre die Frage: Wann liegen welche relevanten Zustandsinformationen dem Steuersystem vor?

In unserem Aufzugsystem sind zukünftige Aufträge nicht mit Sicherheit vorhersagbar. Es ist nun wichtig, ob man belastbare stochastische Informationen bekommen kann oder nicht.

Falls ja, so ist eine Modellierung als *Markovsches Entscheidungsproblem* prinzipiell möglich. Gesucht ist dann eine *optimale Politik*, eine vorausberechnete Funktion, die aus einer Zustandsbeobachtung eine Steuerentscheidung macht, so dass die erwarteten Durchschnittskosten (oder die diskontierten Gesamtkosten, oder ...) minimal sind.

Wir berichten über unser Aufzugsteuerungsproblem: Es ist keine optimale Politik bekannt. Es ist nicht einmal bekannt, wie weit die Kosten heuristischer Politiken von den optimalen Kosten ungefähr abweichen. Genau genommen ist gar nichts Theoretisches über Politiken für das Aufzugsteuerungsproblem bekannt.

Die Beobachtungen im Versandlager Falkensee ließen aber sowieso kein Muster für ankommende Paletten erkennen. Kein Experte hatte eine Idee, wie ein Ankunftsprozess von Paletten eines Auftragstyps aussehen könnte. Nicht alles, was man nicht kennt, kann man unbedingt auch durch Zufallseinfluss beschreiben. Hier war jedenfalls kein belastbares stochastisches Modell zu

finden, das z. B. auch das Eintreffen einer Lieferung mit Papier auf 20 Paletten abdeckt, die alle auf einmal von der zweiten in die fünfte Etage wollen. Unser Entschluss: Wir benutzen keine Information über zukünftige Aufträge. Damit ist unser Steuerproblem ein sogenanntes *Online-Optimierungsproblem*.

In diesem Falle besteht eine gängige Modellierung in der Formulierung als *Frage-Antwort-Spiel*. Die Fragen sind Aufträge, die Antworten sind Steuerentscheidungen eines *Online-Algorithmus*. Die vorherrschende Analysemethode ist die *kompetitive Analyse*. Ein Algorithmus ist salopp gesprochen *c-kompetitiv*, wenn für jede mögliche endliche Folge von Aufträgen seine Gesamtkosten höchstens c-mal so hoch sind wie die eines fiktiven *Offline-Algorithmus*, der die Zukunft vorhersehen kann.

Wir berichten über unser Aufzugsteuerungsproblem: Es ist nicht nur kein kompetitiver Online-Algorithmus für das Aufzugsteuerproblem bekannt, sondern man weiß, dass es keinen kompetitiven Algorithmus dafür gibt. In Ermangelung anderer rigoroser Analysemethoden heißt das, wir sind bzgl. der Online-Problematik auf heuristische Überlegungen angewiesen. (Die sogenannte *Analyse unter vertretbarer Belastung* [HKR00, Ram05] führt auf ähnliche Teilprobleme wie unsere heuristische Überlegung und wird hier einmal beiseite gelassen.)

Eine *Online-Heuristik* (und gleichzeitig eine *Politik*, die etwaige stochastische Informationen ignoriert) ist die sogenannte *Reoptimierungspolitik*, die sich immer dann anbietet, wenn man annehmen kann, dass in einer guten Steuerung neue Aufträge nicht beliebig vorgezogen werden sollten (siehe [GKR$^+$01b, GKR01a, HKR06, FR06] für weitere Online-Heuristiken). Bei der Minimierung von quadratischen Wartezeiten, die für alte Aufträge ja stärker ansteigen, ist das gegeben. Die Reoptimierungspolitik berechnet bei jedem relevanten Ereignis (z. B. Eintreffen eines neuen Auftrags) aus allen zu diesem Zeitpunkt bekannten Daten einen *Plan* aus Steuerentscheidungen, die optimal wären, wenn keine Aufträge mehr kämen. Dieses Optimierungsproblem benutzt als Eingabedaten also einen *Schnappschuss* des Systems und wird daher oft *Schnappschussproblem* genannt.

Ab jetzt nehmen wir an, dass wir für unser Aufzugsystem die Reoptimierungspolitik anwenden. Damit haben wir ein dynamisches Steuerproblem auf ein statisches Optimierungsproblem oder *Offline-Optimierungsproblem* zurückgeführt.

Bevor wir überlegen, wie das zugehörige Schnappschussproblem aussieht, halten wir fest, dass wir in unserem System nicht beliebig lange rechnen sollten, denn während wir rechnen, läuft die Systemzeit weiter. Dies ist ein großer Unterschied zu strategischen Planungsberechnungen, die vor dem operativen Betrieb stattfinden: In Online-Systemen ist es also i. d. R. erforderlich, die Steuerung in Echtzeit zu ermitteln. Wie groß die Antwortzeit höchstens sein darf, begrenzt oft die Auswahl anwendbarer Verfahren.

Modellierungsprozess 1.2.3
Kläre die Frage: Wie schnell muss eine Steuerentscheidung verfügbar sein?

Wenn ein Aufzug frei wird und wir den nächsten zu bearbeitenden Auftrag bestimmen, so bedeutet mehr Rechenzeit einen wartenden Aufzug. Nun kann der frei werdende Aufzug noch vorhergesehen werden, die neu eintreffende Palette aber nicht. Wir nehmen an, dass wir die Zuweisung einer neuen Palette auf einen Aufzug und ebenso die Bestimmung des jeweils nächsten Auftrags für die Aufzüge in etwa einer Sekunde vornehmen müssen.

Damit ist unser Steuerproblem auch ein *Echtzeit-Optimierungsproblem* mit einer Sekunde Antwortzeit. Das bedeutet, bei Anwendung der Reoptimierungspolitik müssen wir in einer Sekunde das Schnappschussproblem ausreichend gut lösen können.

Das gleiche Programm kann für die Gelben Engel durchgeführt werden: Auch hier ist das System gekennzeichnet durch die Daten der Einheiten sowie die Zeitpunkte und Positionen ihrer frühesten Verfügbarkeit und die Daten der wartenden Aufträge. Bei den Kosten dominieren neben Fahrtkosten ebenfalls quadratisch anzusetzende Verspätungskosten (lassen wir Vertragspartner einmal weg). Stochastische Informationen über zukünftige Aufträge ist praktisch nicht bekannt, es handelt sich also wieder um ein Online-Problem. Die Anwendung der Reoptimierungspolitik verlangt diesmal eine Antwortzeit von höchstens zehn Sekunden. Durch Absolvierung dieses System-Modellierungs-Programms wird die Verwandtschaft beider Probleme schon deutlicher.

Aber wie funktioniert nun die erfolgreiche Modellierung und Reoptimierung des statischen Schnappschussproblems genau?

1.3 Zwei Einsatzplanungsmodelle für das statische Schnappschussproblem

Die Modellierung der in Reoptimierungspolitiken (und sonstwo) auftauchenden statischen Optimierungsprobleme funktioniert meist in denselben Schritten wie in Abschnitt 1.2, nur ohne die dynamische Komponente und mit vollständiger Information. Das ist der Vorteil der Reoptimierungspolitik.

Modellierungsprozess 1.3.1
Spezifiziere die verfügbaren Informationen (also den momentan vorliegenden beobachtbaren Systemzustand) für ein Schnappschussproblem durch abstrakte Strukturen und Eingabedaten, Entscheidungsoptionen durch Variablen, unmögliche Entscheidungen durch Nebenbedingungen und die Qualität einer Entscheidung durch eine Kostenfunktion der Variablen.

Die logische Struktur eines Problems (als Teil der verfügbaren Informationen) wird meistens gewinnbringend durch abstrakte (meist graphentheoretische) Strukturen abgebildet. In diesem Abschnitt modellieren wir das statische Schnappschussproblem beim ADAC durch ein abstraktes Transportnetz.

Die Modellierung wollen wir zur Vereinfachung in diesem Abschnitt nur für lineare Verspätungskosten behandeln. Wir werden sehen, dass nur eines der beiden vorgeschlagenen Modelle eine einfache Erweiterung auf nicht-lineare Verspätungskosten erlaubt.

Spezifizieren wir zunächst allgemein die beobachtbaren Daten, die den Systemzustand des Schnappschussproblems beschreiben. Es sei dazu $\mathscr{E}_c$ die Menge aller aktuell bekannten Aufträge und

$$\mathscr{E}_c^u := \{e \in \mathscr{E}_c \mid F_e \subseteq F_u\} \tag{1.1}$$

die Menge aller aktuell bekannten Aufträge, die von Einheit u bearbeitet werden können (F_e ist eine Menge von Anforderungen des Auftrags e, und F_u ist eine Menge von Fähigkeiten von Einheit u, z. B. Starthilfe, Abschleppen, ...). Wir konstruieren nun einen gerichteten Graphen – das ist der *Transportgraph* $G_u = (V_u, E_u)$ mit Knotenmenge

$$V_u := \mathscr{E}_c^u \cup \{o_u\} \cup \{d_u\}, \tag{1.2}$$

wobei o_u die Position der frühesten Verfügbarkeit und d_u die Heimatposition von u seien (in GPS-Koordinaten gemessen).

Der Transportgraph sei vollständig bezüglich der Aufträge, die Position o_u hat eine ausgehende Kante mit jedem anderen Knoten und die Position d_u eine eingehende. Weiterhin existieren keine Schleifen, d. h. die Kantenmenge ist

$$E_u := \left\{ \mathscr{E}_c^u \times \mathscr{E}_c^u \backslash \{(e,e) \mid e \in \mathscr{E}_c^u\} \right\} \cup \left\{ (\{o_u\} \times \mathscr{E}_c^u) \cup (\mathscr{E}_c^u \times \{d_u\}) \right\}. \tag{1.3}$$

Die Fahrzeit von Einheit u von Knoten i zu Knoten j sei $\delta_u^{i,j}$; die Servicezeit bei Auftrag i sei δ_i; t_u^{start} bezeichne den Zeitpunkt der frühesten Verfügbarkeit von Einheit u (denn u könnte dann Schichtbeginn haben oder aber gerade einen Auftrag bearbeiten), und t_u^{end} sei der Zeitpunkt des Schichtendes von Einheit u. Zur Vereinheitlichung setzen wir $\delta_{o_u} := t_u^{\text{start}}$ sowie $\delta_{d_u} := 0$. Zur Abkürzung sei ferner $\tau_u^{i,j} := \delta_u^{i,j} + \delta_i$ die Mindestzeit zwischen den Service-Startzeiten bei Knoten i und j. Es bezeichne t_v^e den Zeitpunkt, zu dem Vertragspartner v bei Auftrag e sein kann. Beim ADAC gibt jeder Vertragspartner diese Werte für alle Aufträge innerhalb eines bestimmten Gebietes an.

Darüber hinaus haben wir eine früheste Service-Startzeit Θ_j^r (*Release-Zeit*) und eine späteste (verspätungskostenlose) Service-Startzeit Θ_j^d (*Deadline*) für jeden Auftrag j. Wir setzen $\Theta_{o_u}^r := \Theta_{d_u}^r := 0$, um für jeden Knoten j im Transportgraphen ein definiertes Θ_j^r und Θ_j^d zu haben.

Die Kosten pro Zeiteinheiten Fahren, Service, Überstunden und Verspätung seien c_u^{drv}, c_u^{srv}, c_u^{ovt} bzw. c_e^{late}. (Hier steckt die Annahme linearer Verspätungskosten und Überstundenkosten drin.)

Den Transportgraph zusammen mit allen diesen Daten nennen wir kurz *Transportnetz* des Problems.

Wir präsentieren nun zwei mögliche Modelle für das Einsatzplanungsproblem in unserem Transportnetz: eines hat viele Nebenbedingungen, eines hat sehr viele Variablen. Es wird sich herausstellen, dass nur eines von beiden zum Erfolg führt. Obwohl wir im Folgenden uns sprachlich gelegentlich zur Veranschaulichung auf das ADAC-Problem beziehen, sollte klar werden, dass man die folgenden Modelle allein mit den Objekten des Transportnetzes ausdrücken könnte. Wir befinden uns also bereits in einer *Abstraktion* des Problems.

1.3.1 Modellierung als MINCOSTFLOW

Zur Erinnerung: Wir brauchen bei gegebener momentaner Auftragslage Touren für jedes Fahrzeug, die die Aufträge partitionieren. Die Partitions- und Tourenbestimmung wollen wir für lineare Verspätungskosten in diesem Abschnitt als ein geschlossenes ganzzahliges lineares Flussproblem – genauer als *Multi-Commodity-Flow-Problem mit weichen Zeitfenstern* – formulieren.

Wir behandeln zuerst Entscheidungen, Nebenbedingungen und Kosten für das Tourenproblem und fügen dann die Nebenbedingungen für das Partitionsproblem hinzu.

Beginnen wir mit dem Tourenproblem. Eine Tour für Einheit u ist eine Tour im Transportgraphen G_u. Die zur Bestimmung einer solchen Tour notwendigen Entscheidungen kann man als von Einheit u zu durchfahrende Kanten in G_u auffassen. Damit ist die Entscheidung für eine Tour darstellbar als Auswahl einer Kantenmenge in G_u, die einer zulässigen Tour für u entspricht. Sei $y_u^{i,j}$ eine Variable mit Werten in $\{0,1\}$ (eine *Binärvariable*). Wenn $y_u^{i,j}$ den Wert Eins annimmt,

dann interpretieren wir das als Auswahl der Kante (i, j) für die Tour von Einheit u. So werden unsere Entscheidungen durch *unabhängige Variablen* $y_u^{i,j}$ für die Tourenplanung modelliert.

Kommen wir zu den Nebenbedingungen. Da Einheit u jeden Auftrag, den sie anfährt, auch wieder verlassen muss, müssen die Variablenbelegungen für $y_u^{i,j}$ den *Flussbedingungen* genügen. Die Tour startet an der augenblicklichen Position und endet an der Heimatposition. Das ergibt folgende Nebenbedingungen:

$$\sum_{j:(j,i)\in E_u} y_u^{j,i} - \sum_{j:(i,j)\in E_u} y_u^{i,j} = \begin{cases} 0 & i \in \mathscr{E}_c^u \\ -1 & i = o_u \\ 1 & i = d_u \end{cases} \tag{1.4}$$

Eine Lösung des bisher Beschriebenen ist ein *binärer Fluss* von o_u nach d_u.

Allerdings sind sogenannte *Subtouren* noch nicht ausgeschlossen. Dies bedeutet, dass der Fluss in einen Weg von o_u nach d_u (erwünscht) und ein oder mehrere Zyklen (unerwünscht) zerfallen kann. Außerdem können wir bisher die anfallenden Verspätungs- und Überstundenkosten nicht bestimmen.

Zu diesem Zweck führen wir *abhängige Variablen* t_u^i für jeden Knoten an, die jeweils den Zeitpunkt des Servicebeginns von Einheit u bei Knoten i angeben.

Berücksichtigen wir noch die Release-Zeit Θ_j^r, so bekommen wir für den Beginn des Services bei j

$$t_u^j = \max\{t_u^i + \tau_u^{i,j}, \Theta_j^r\}, \tag{1.5}$$

falls u direkt von i nach j fährt. Wir formulieren dies folgendermaßen als Nebenbedingung:

$$y_u^{i,j}(\max\{t_u^i + \tau_u^{i,j}, \Theta_j^r\} - t_u^j) = 0. \tag{1.6}$$

Diese Bedingung ist nicht linear, wir können sie aber durch einen gängigen Trick in eine lineare Bedingung überführen. Dazu ersetzen wir zuerst $= 0$ durch ≤ 0. Dies wird dadurch gerechtfertigt, dass durch Vergrößern von t_u^j die Kosten nicht gesenkt werden können. Jetzt wenden wir die sogenannte *Big-M-Methode* an: Für jeden Auftrag $e \in \mathscr{E}_c^u$ sei $\hat{t}_u^e$ eine obere Schranke für den Wert von t_u^e in einer Optimallösung und $\check{t}_u^e$ eine untere. Es sei M so groß gewählt, dass für je zwei Aufträge i und j für den Abstand $|\hat{t}_u^i + \tau_u^{i,j} - \check{t}_u^j| \leq M$ gilt. Dann können wir Bedingung (1.6) ersetzen durch:

$$t_u^i + \tau_u^{i,j} - t_u^j \leq M(1 - y_u^{i,j}) \tag{1.7}$$

$$\Theta_j^r \leq t_u^j \tag{1.8}$$

Diese Bedingung verhindert nun die Bildung von Subtouren, da die $\tau_u^{i,j}$ auf Grund der Daten alle streng positiv sind. Dies wäre auch anders möglich gewesen, aber die Variablen t_u^i werden sowieso zur Berechnung der Verspätungskosten gebraucht.

Kommen wir zu den Kosten. Die der Kante (i, j) unmittelbar zuzuordnenden Kosten sind die Fahrtkosten $c_u^{\text{drv}}\delta_u^{i,j}$. Zusätzlich gibt es noch Kosten, die am Knoten j entstehen, nämlich die Servicekosten $c_u^{\text{srv}}\delta_j$. Erinnerung: $\delta_{o_u} := t_u^{\text{start}}$ und $\delta_{d_u} := 0$. Um zusätzliche Variablen zu

vermeiden, weisen wir die Servicekosten immer den Endknoten einer Kante zu, d. h. die Kosten $c_u^{i,j}$ der Kante $(i,j) \in E_u$ sind

$$c_u^{i,j} := c_u^{\text{drv}} \delta_u^{i,j} + c_u^{\text{srv}} \delta_j \tag{1.9}$$

Diese Kosten sind damit *asymmetrische Kantenkosten*. Wir haben bei den $c_u^{i,j}$ die Kosten des Endknotens der Kante, aber bei $\tau_u^{i,j}$ die Servicezeit beim Anfangsknoten verwendet. Dies erklärt sich folgendermaßen: Befindet sich Einheit u beim Knoten i, so entstehen durch die Entscheidung, zu Knoten j zu fahren, unmittelbar die Fahrtkosten für den Weg von i zu j und die Servicekosten bei j. Der Service bei Knoten i ist bereits beendet und unabhängig von der Wahl des nächsten Knotens. Bei der Zeitberechnung ist es genau umgekehrt, da wir die Anfangszeiten des Services betrachten. Um festzustellen, wann der Service bei j beginnen kann, müssen wir wissen, wann er bei i begonnen hat, wie lange er dort dauerte und wie viel Zeit für die Fahrt zu j benötigt wird. Die Servicedauer bei Knoten j ist für den Beginnzeitpunkt unbedeutend.

Um nun die Verspätungs- und Überstundenkosten bestimmen zu können, führen wir zusätzliche *abhängige Variablen* $t_{u,e}^l$ für $e \in \mathscr{E}_c^u$ (Verspätung von u bei e) für Verspätung sowie t_u^o (Schichtzeitüberschreitung von u) für Überstunden ein. Damit diese wirklich die Verspätung messen, müssen folgende Ungleichungen gelten:

$$t_{u,e}^l \geq 0 \tag{1.10}$$

$$t_{u,e}^l \geq t_u^e - \Theta_i^d, e \in \mathscr{E}_c^u \tag{1.11}$$

$$t_u^o \geq 0 \tag{1.12}$$

$$t_u^o \geq t_u^{d_u} - t_u^{\text{end}} \tag{1.13}$$

Diese Variablen werden dann in der Zielfunktion mit den Kosten pro Zeiteinheit Verspätung c_e^{late} bzw. c_u^{ovt} gewichtet. Hier geht wieder ein, dass wir nur lineare Verspätungskosten modellieren wollen.

Die Variable $t_u^{o_u}$ kann dann mit 0 fest vorbelegt werden, da die früheste Verfügbarkeit von u ja schon durch $\delta_{o_u} = t_u^{\text{start}}$ berücksichtigt ist.

Für Vertragspartner $v \in \mathscr{V}$ muss nur die Menge der ihm zugewiesenen Aufträge bestimmt werden und keine Tour. Analog zu den Einheiten setzen wir

$$\mathscr{E}_c^v := \{e \in \mathscr{E}_c \mid F_e \subseteq F_v\} \tag{1.14}$$

und führen für jeden Auftrag $e \in \mathscr{E}_c^v$ die unabhängige binäre Variable y_v^e ein, die angibt, ob Vertragspartner v Auftrag e bearbeitet ($y_v^e = 1$) oder nicht ($y_v^e = 0$). Die Kosten für die Fremdvergabe von Auftrag e zu Vertragspartner v lauten

$$c_v^{\text{srv}} + c_e^{\text{late}} \max\{t_v^e - \Theta_e^d, 0\} \tag{1.15}$$

Dabei ist zu beachten, dass t_v^e keine Optimierungsvariable, sondern ein vom Vertragspartner genannter Parameter ist.

Damit haben wir Touren separat für jede Einheit so modelliert, dass wir bei gegebenen unabhängigen Entscheidungsvariablen die davon verursachten Kosten über den Umweg weiterer abhängiger Hilfsvariablen messen können.

Nun muss die Partitionierung der Aufträge in Touren beachtet werden: Eine Lösung der bisherigen Nebenbedingungen berechnet zwar eine Tour für jede Einheit, stellt aber dabei noch nicht sicher, dass jeder Auftrag von genau einer Einheit oder genau einem Vertragspartner bearbeitet wird. Wird Auftrag e von Vertragspartner v bearbeitet, so ist $y_v^e = 1$. Bei Einheiten ist dies schwieriger festzustellen: Einheit u muss in diesem Fall über genau eine Kante $(j,e) \in E_u$ zu Auftrag e fahren. Dies bedeutet

$$\sum_{j:(j,e)\in E_u} y_u^{j,e} = 1 \tag{1.16}$$

Umgekehrt darf Einheit u nur dann von Knoten j zu Auftrag e fahren, wenn dieser direkt nach j von u bearbeitet wird. Damit schreibt sich die Mengenpartitionsbedingung für Auftrag $e \in \mathscr{E}_c$ folgendermaßen:

$$\sum_{u:e\in\mathscr{E}_c^u} \sum_{j:(j,e)\in E_u} y_u^{j,e} + \sum_{v:e\in\mathscr{E}_c^v} y_v^e = 1 \tag{1.17}$$

Wir bekommen ein gemischt ganzzahliges lineares Optimierungsproblem, das zum Beispiel mit Branch-and-Bound-Methoden, wie sie Standard MIP-Löser verwenden, gelöst werden kann:

Problem 1.3.2 (IFP)

$$\begin{aligned}
\text{Minimiere} \quad & \sum_{u\in\mathscr{U}} \sum_{(i,j)\in E_u} c_u^{i,j} y_u^{i,j} \\
& + \sum_{e\in\mathscr{E}_c} \sum_{u\in\mathscr{U}:e\in\mathscr{E}_c^u} c_e^{\text{late}} t_{u,e}^l \\
& + \sum_{e\in\mathscr{E}_c} \sum_{v\in\mathscr{V}:e\in\mathscr{E}_c^v} y_v^e \left(c_v^{\text{srv}} + c_e^{\text{late}} \max\{t_v^e - \Theta_e^d, 0\}\right) \\
& + \sum_{u\in\mathscr{U}} c_u^{\text{ovt}} t_u^o
\end{aligned} \tag{1.18}$$

unter den Nebenbedingungen

$$\sum_{u:e\in\mathscr{E}_c^u}\sum_{j:(j,e)\in E_u} y_u^{j,e} + \sum_{v:e\in\mathscr{E}_c^v} y_v^e = 1 \qquad \forall\, e\in\mathscr{E}_c \tag{1.19}$$

$$\sum_{j:(j,e)\in E_u} y_u^{j,e} - \sum_{j:(e,j)\in E_u} y_u^{e,j} = 0 \qquad \forall\, u\in\mathscr{U}\ \forall\, e\in\mathscr{E}_c^u \tag{1.20}$$

$$\sum_{j:(o_u,j)\in E_u} y_u^{o_u,j} = 1 \qquad \forall\, u\in\mathscr{U} \tag{1.21}$$

$$\sum_{j:(j,d_u)\in E_u} y_u^{j,d_u} = 1 \qquad \forall\, u\in\mathscr{U} \tag{1.22}$$

$$t_u^i + \tau_u^{i,j} - t_u^j - M(1-y_u^{i,j}) \le 0 \qquad \forall\, u\in\mathscr{U}\ \forall\,(i,j)\in E_u \tag{1.23}$$

$$t_u^e \ge 0 \qquad \forall\, u\in\mathscr{U}\ \forall\, e\in\mathscr{E}_c^u \tag{1.24}$$

$$t_u^{d_u} \ge 0 \qquad \forall\, u\in\mathscr{U} \tag{1.25}$$

$$t_u^e \ge \Theta_e^r \qquad \forall\, u\in\mathscr{U}\ \forall\, e\in\mathscr{E}_c^u \tag{1.26}$$

$$t_u^e - \Theta_e^d - t_{u,e}^l \le 0 \qquad \forall\, u\in\mathscr{U}\ \forall\, e\in\mathscr{E}_c^u \tag{1.27}$$

$$t_{u,e}^l \ge 0 \qquad \forall\, u\in\mathscr{U}\ \forall\, e\in\mathscr{E}_c^u \tag{1.28}$$

$$t_u^{d_u} - t_u^{\text{end}} - t_u^o \le 0 \qquad \forall\, u\in\mathscr{U} \tag{1.29}$$

$$t_u^o \ge 0 \qquad \forall\, u\in\mathscr{U} \tag{1.30}$$

$$y_u^{i,j} \in \{0,1\} \qquad \forall\, u\in\mathscr{U}\ \forall\,(i,j)\in E_u \tag{1.31}$$

$$y_v^e \in \{0,1\} \qquad \forall\, v\in\mathscr{V}\ \forall\, e\in\mathscr{E}_c^v \tag{1.32}$$

1.3.2 Das Tourenmodell

Wir modellieren die Touren und Partitionierung wieder in zwei Schritten. Die Tourenmodellierung wird gewissermaßen mit der Keule durchgeführt: Wir nehmen an, wir hätten irgendwo eine Menge $\mathscr{R}$ mit allen zulässigen Touren; $\mathscr{R}_u$ bezeichne die Teilmenge der für u zulässigen Touren. Nehmen wir eine Tour R aus dieser Menge her. Wir definieren die *unabhängigen Variablen* x_R mit Werten in $\{0,1\}$ wie folgt: Wenn x_R den Wert Eins annimmt, dann heißt das, Tour R wird benutzt und umgekehrt.

Die Tour R ist durch die Angabe der Bearbeitungsreihenfolge $e_1,\ldots,e_k$ und der Einheit u vollständig bestimmt.

Nebenbedingungen für die Zusammensetzung einer Tour aus einzelnen Reihenfolgeentscheidungen gibt es nun plötzlich nicht mehr.

Die Kosten dieser Tour sind

$$\begin{aligned} c_R := {} & c_u^{\mathrm{drv}} \delta_u^{o_u,e_1} + \sum_{i=1}^{k-1} c_u^{\mathrm{drv}} \delta_u^{e_i,e_{i+1}} + c_u^{\mathrm{drv}} \delta_u^{e_k,d_u} \\ & + \sum_{i=1}^{k} c_u^{\mathrm{srv}} \delta_{e_i} \\ & + \sum_{i=1}^{k} c_{e_i}^{\mathrm{late}} \max\{t_u^{e_i} - \Theta_{e_i}^d, 0\} \\ & + c_u^{\mathrm{ovt}} \max\{t_u^{d_u} - t_u^{\mathrm{end}}, 0\} \end{aligned} \tag{1.33}$$

Beachte, dass es nun im Prinzip nichts ausmachte, wenn die Verspätungskosten eine nichtlineare Funktion der Verspätungszeiten wären.

Für die Vertragspartner sei $\mathscr{S}$ die Menge aller elementaren Zuweisungen, d. h. ein $S \in \mathscr{S}$ gehört zu genau einem Vertragspartner $v \in \mathscr{V}$ und zu genau einem Auftrag $e \in \mathscr{E}_c^v$ und entspricht der Zuweisung von e zu v. Die Kosten dieser Entscheidung S sind dann

$$c_S := c_v^{\mathrm{srv}} + \left\{t_v^e - \Theta_e^d, 0\right\} \tag{1.34}$$

Nun fehlen die Partitionierungs-Bedingungen, die wie folgt modelliert werden: definiere die *Inzidenzmatrix* von Aufträgen und Touren durch $a_{Re} := 1$ genau dann, wenn Auftrag e in Tour R vorkommt, $a_{Re} := 0$ sonst. Analog sei $b_{S_e} := 1$ genau dann, wenn Auftrag e zur Zuweisung S gehört. Dann bedeutet die Bedingung, dass jeder Auftrag in genau einer Tour oder in genau einer Zuweisungsmenge für Vertragspartner vorkommt:

$$\sum_{S \in \mathscr{S}} b_{Se} x_S + \sum_{R \in \mathscr{R}} a_{Re} x_e = 1 \quad \forall\, e \in \mathscr{E}_c \tag{1.35}$$

Die Tatsache, dass jede Einheit genau eine Tour fährt, schreibt sich dann:

$$\sum_{R \in \mathscr{R}} a_{Ru} x_R = 1 \quad \forall\, u \in \mathscr{U} \tag{1.36}$$

Das nun rein ganzzahlige Modell lautet jetzt übersichtlich:

Problem 1.3.3 (ITMP)

$$\text{Minimiere} \sum_{S \in \mathscr{S}} c_S x_S + \sum_{R \in \mathscr{R}} c_R x_R \tag{1.37}$$

unter den Nebenbedingungen

$$\sum_{S \in \mathscr{S}} b_{Se} x_S + \sum_{R \in \mathscr{R}} a_{Re} x_e = 1 \qquad \forall\, e \in \mathscr{E}_c \tag{1.38}$$

$$\sum_{R \in \mathscr{R}} a_{Ru} x_R = 1 \qquad \forall\, u \in \mathscr{U} \tag{1.39}$$

$$x_R \in \{0,1\} \qquad \forall\, R \in \mathscr{R} \tag{1.40}$$

$$x_S \in \{0,1\} \qquad \forall\, S \in \mathscr{S} \tag{1.41}$$

Wie haben wir das geschafft? Wir können mit diesem Modell nichtlineare Verspätungskosten modellieren und haben nur noch Partitionsbedingungen im Modell – alles sieht viel einfacher aus. Nun: Wir haben unter den Teppich gekehrt, dass das Tourenmodell unglaublich viele Variablen hat: soviele, wie es zulässige Touren gibt. Und das sind astronomisch viele. Für unsere Standard-Problemskalen (200 Aufträge, 100 Einheiten) sind das mehr Touren und damit mehr Variablen als Atome im Universum. Man kann das Modell also nicht einfach mit einem kommerziellen Löser verarbeiten, da man noch nicht einmal das Problem generieren und einlesen kann. Zu Lösungsverfahren kommen wir im folgenden Abschnitt.

Aber was ist mit dem Aufzugsystem? Wir können im Prinzip die identischen Modelle betrachten. Die einzige dazu notwendige logische Überlegung ist, wie ein relevanter Systemzustand des Aufzugsystems mit dem gleichen Datenmodell wie oben als Transportnetz dargestellt werden kann. Knoten des Transportnetzes entsprechen wieder Aufträgen und die Start- und Endposition der Einheiten sind in diesem Falle Aufzüge. Wir beobachten noch, dass eine Wartezeit gleich einer Verspätungszeit bzgl. $\Theta_j^d = \Theta_j^r$ ist, wobei Θ^r im Falle der Aufzüge der Zeitpunkt des Eintreffens von Auftrag j ist.

Es scheint nun ein Unterschied zum ADAC-Problem dadurch zu entstehen, dass sich beim Bearbeiten eines Transportauftrags die Position des Servers ändert (die lässt sich damit vergleichen, dass eine ADAC-Einheit abschleppen muss). Das kann uns aber ganz und gar egal sein, denn für das den obigen Modellen zugrunde liegende abstrakte Transportnetz sind nur die Übergangszeiten vom Beginn eines Auftrags zum Beginn des nächsten interessant. Und die sind beim Aufzug gegeben durch die Dauer der Transportfahrt für einen Auftrag plus die Dauer der Fahrt vom Transportziel des einen Auftrags zum Transportbeginn des nächsten Auftrags. Die Transportdauer beim Aufzug entspricht der Servicezeit beim ADAC. Leerfahrtzeiten sind zwar asymmetrisch im Gegensatz zu reinen Fahrzeiten beim ADAC, aber die Gesamtübergangszeiten sind beim ADAC auch asymmetrisch wegen der Verbuchung der Servicezeiten; daher richten die asymmetrischen Leerfahrtzeiten keinen Schaden mehr an.

Zu beachten ist allerdings, dass Aufträge in der lokalen Warteschlange vor einem Aufzug nicht mehr einem anderen Aufzug zugeordnet werden können. Damit sind alle Touren, die diesen Auftrag mit einem anderen Aufzug befördern, automatisch unzulässig. Das wird im Flussmodell durch Null-Fixierung von Flussvariablen und im Tourenmodell durch Ausschluss entsprechender Touren aus der Menge aller zulässigen Touren erreicht.

Hier sehen wir deutlich die Stärke des Abstraktionsschrittes über den Umweg durch die Graphentheorie: Jedes Einsatzplanungsproblem, das als ein Einsatzplanungsproblem in einem Transportnetz wie oben formuliert werden kann, wird durch die oben besprochenen Modelle korrekt formuliert.

Im nächsten Abschnitt wollen wir zeigen, wie man die Modelle lösen kann.

1.4 Lösungsverfahren für die Modelle

Das Flussmodell ist klein genug, um es einem kommerziellen MIP-Löser zu übergeben. Man kann diesen Umstand verstehen als die Umsetzung der mittlerweile weit entwickelten Theorie der gemischt-ganzzahligen Programmierung. Das ist mathematisch etwas langweilig, aber häufig sehr nützlich.

Das Tourenmodell allerdings erfordert Handarbeit: Wie bereits erwähnt besitzt das Tourenmodell 1.3.3 astronomisch viele Touren; zu viele, um sie in einem heutigen Computer auch nur speichern zu können. Die Kenntnis all dieser Touren ist jedoch gar nicht nötig. Uns würden im Prinzip die Touren reichen, die in einer kostengünstigsten Lösung von den Einheiten tatsächlich gefahren werden. Da wir diese im Vorfeld nicht kennen, suchen wir zuerst mit Hilfe von Heuristiken eine gute Startlösung. Hier zeigt sich ein entscheidender Vorteil dieses Ansatzes gegenüber der Flussmodellierung: Wenn wir die Rechnung wegen der knappen Rechenzeit abbrechen müssen, so haben wir zumindest einen gültigen Einsatzplan bestimmt. Diesen Plan wollen wir dann sukzessive verbessern bzw. feststellen, dass keine Verbesserung mehr möglich ist. Dieses Verfahren wird *dynamische Spaltengenerierung* genannt. Um dies in der Praxis durchzuführen, müssen wir unser Problem zuerst vereinfachen. Das *Tourenmaster-Problem* lautet:

Problem 1.4.1 (TMP)

$$\text{Minimiere} \sum_{S\in\mathscr{S}} c_S x_S + \sum_{R\in\mathscr{R}} c_R x_R \tag{1.42}$$

unter den Nebenbedingungen

$$\sum_{S\in\mathscr{S}} b_{Se} x_S + \sum_{R\in\mathscr{R}} a_{Re} x_e = 1 \qquad \forall\, e \in \mathscr{E}_c \tag{1.43}$$

$$\sum_{R\in\mathscr{R}} a_{Ru} x_R = 1 \qquad \forall\, u \in \mathscr{U} \tag{1.44}$$

$$x_R \geq 0 \qquad \forall\, R \in \mathscr{R} \tag{1.45}$$

$$x_S \geq 0 \qquad \forall\, S \in \mathscr{S} \tag{1.46}$$

Wir haben bei Problem 1.4.1 auf die Forderung $x_R, x_S \in \{0,1\}$ verzichtet. Damit bekommen wir potenziell mehr Lösungen und das Minimum wird höchstens billiger. Wir erhalten also *unvermeidbare Kosten*.

Beginnen wir mit der Lösung von 1.4.1: Für jeden Vertragspartner gibt es nur die elementaren Entscheidungen „Auftrag nehmen oder nicht“. Die Menge $\mathscr{S}$ kann also vorab generiert werden. Bei den Touren der Einheiten sieht es anders aus, da hier die Reihenfolge der Aufträge relevant ist. Bezeichnet nun $\widetilde{\mathscr{R}}$ die Menge aller Touren, die bereits bekannt sind, so lautet das *reduzierte Tourenmaster-Problem*:

Problem 1.4.2 (RTMP)

$$\text{Minimiere} \sum_{S\in\mathscr{S}} c_S x_S + \sum_{R\in\widetilde{\mathscr{R}}} c_R x_R \tag{1.47}$$

unter den Nebenbedingungen

$$\sum_{S\in\mathscr{S}} b_{Se} x_S + \sum_{R\in\widetilde{\mathscr{R}}} a_{Re} x_e = 1 \qquad \forall\, e \in \mathscr{E}_c \tag{1.48}$$

$$\sum_{R\in\widetilde{\mathscr{R}}} a_{R_u} x_R = 1 \qquad \forall\, u \in \mathscr{U} \tag{1.49}$$

$$x_R \geq 0 \qquad \forall\, R \in \widetilde{\mathscr{R}} \tag{1.50}$$

$$x_S \geq 0 \qquad \forall\, S \in \mathscr{S} \tag{1.51}$$

Dieses Problem können wir nun mit Hilfe von Verfahren der linearen Programmierung, z. B. dem *Simplexalgorithmus*, lösen. Die Optimallösung $\hat{x}$ von 1.4.2 ist auf jeden Fall eine gültige Lösung

für 1.4.1. Aber ist sie auch kostenminimal? Um diese Frage beantworten zu können, müssen wir uns dem dualen Problem zuwenden. Das *duale reduzierte Tourenmaster-Problem*:

Problem 1.4.3 (DRTMP)

$$\text{Maximiere} \sum_{e \in \mathscr{E}_c} \pi_e + \sum_{u \in \mathscr{U}} \pi_u \tag{1.52}$$

unter den Nebenbedingungen

$$\sum_{e \in \mathscr{E}_c} b_{Se} \pi_e \leq c_S \qquad \forall\, S \in \mathscr{S} \tag{1.53}$$

$$\sum_{e \in \mathscr{E}_c} a_{Re} \pi_e + \sum_{u \in \mathscr{U}} a_{R_u} \pi_u \leq c_R \qquad \forall\, R \in \widetilde{\mathscr{R}} \tag{1.54}$$

Wir sehen: Jede neue Spalte für Problem 1.4.2 entspricht einer neuen Ungleichung im dualen Problem 1.4.3. Nach der Dualitätstheorie der linearen Optimierung gilt: Die zulässige Lösung $\hat{x}$ für 1.4.2 und die zulässige Lösung $\hat{\pi}$ für 1.4.3 sind genau dann Optimallösungen, wenn

$$\sum_{S \in \mathscr{S}} c_S \hat{x}_S + \sum_{R \in \widetilde{\mathscr{R}}} c_R \hat{x}_R = \sum_{e \in \mathscr{E}_c} \hat{\pi}_e + \sum_{u \in \mathscr{U}} \hat{\pi}_u. \tag{1.55}$$

Erfüllt nun $\hat{\pi}$ auch die Ungleichungen bezüglich der noch nicht erzeugten Touren, so ist $\hat{\pi}$ zulässig für das nichtreduzierte duale Tourenmaster Problem und damit gewinnmaximal. Nach (1.55) ist $\hat{x}$ dann eine kostenminimale Lösung von 1.4.1.

Falls es eine noch nicht generierte Tour R für Einheit u gibt mit

$$\sum_{e \in \mathscr{E}_c} a_{Re} \hat{\pi}_e + \hat{\pi}_u > c_R \Leftrightarrow \bar{c}_R := c_R - \sum_{e \in \mathscr{E}_c} a_{Re} \hat{\pi}_e - \hat{\pi}_u < 0, \tag{1.56}$$

so nehmen wir diese Tour hinzu und lösen das reduzierte Problem erneut. Dabei heißen die $\bar{c}_R$ *reduzierte Kosten* und die $\hat{\pi}_e$ *duale Preise*.

Da es in 1.4.2 astronomisch viele Variablen gibt, gibt es in 1.4.3 astronomisch viele Ungleichungen. Alle direkt zu überprüfen wäre ein nicht zu bewältigender Aufwand. Die Idee ist nun die folgende: Es gibt eine solche Tour genau dann, wenn $\min_R \bar{c}_R < 0$ ist. In diesem Fall liefert uns der Minimierer auch gleich eine verletzte duale Ungleichung. Die Bestimmung dieses Minimums kann mit Hilfe von Optimierungsverfahren erheblich schneller erfolgen, als alle Touren zu enumerieren. Ein weiterer Vorteil des Tourenmodells ist, dass dieses *Teilproblem* für jede Einheit separat gelöst werden kann.

Wenn es keine Touren mit negativen reduzierten Kosten $\bar{c}_R$ mehr gibt, so ist das Problem 1.4.1 optimal gelöst. Wir haben damit eine untere Schranke für einen kostengünstigsten Einsatzplan bestimmt. Falls die Lösung von 1.4.1 nicht ganzzahlig ist, so lösen wir das ganzzahlige Problem 1.3.3 mit der Menge der bereits generierten Touren. Dies liefert uns einen gültigen Einsatzplan und damit eine obere Schranke. Bei ADAC-Daten ist die Lücke i. d. R. $\leq 5\%$ [KRT02], so dass man sich damit zufrieden gibt. Will man eine bessere Gütegarantie bekommen, so muss man die Tourengenerierung mit der Lösung des ganzzahligen Programmes verheiraten. Diese sogenannten *Branch-and-Price*-Ansätze erforden aber einen erheblich höheren Rechenaufwand und sind damit für zeitkritische Anwendungen wie ADAC oder Fahrstuhlsteuerung nicht geeignet.

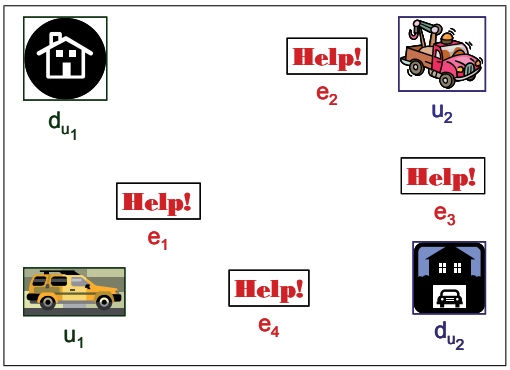


Abbildung 1.1: Beispiel

1.5 Beispiel

Betrachten wir ein kleines Beispiel (siehe Abbildung 1.1). Qualitative Rechenergebnisse für Produktionsdaten finden sich in [KRT02].

Beispiel 1.5.1
Hier gibt es zwei Fahrzeuge u_1 und u_2 sowie vier Aufträge e_1 bis e_4. Der Auftrag e_4 ist dabei zu Beginn noch nicht bekannt und wird deshalb beim ersten Schnappschussproblem nicht berücksichtigt.

Die Entfernungen für die Einheiten u_1 und u_2 seien gegeben durch

	o_{u_1}	d_{u_1}	e_1	e_2	e_3	e_4
o_{u_1}	0	10	5	17,5	22,5	10
d_{u_1}	10	0	7,5	17,5	25	20
e_1	5	7,5	0	12	15	10
e_2	17,5	17,5	12	0	10	17
e_3	22,5	25	15	10	0	15
e_4	10	20	10	17	15	0

	o_{u_2}	d_{u_2}	e_1	e_2	e_3	e_4
o_{u_2}	0	20	22,5	10	5	25
d_{u_2}	20	0	25	20	17,5	15
e_1	22,5	25	0	18	23	17
e_2	10	20	18	0	12,25	20
e_3	5	17,5	23	12,25	0	20
e_4	25	15	17	20	20	0

(1.57)

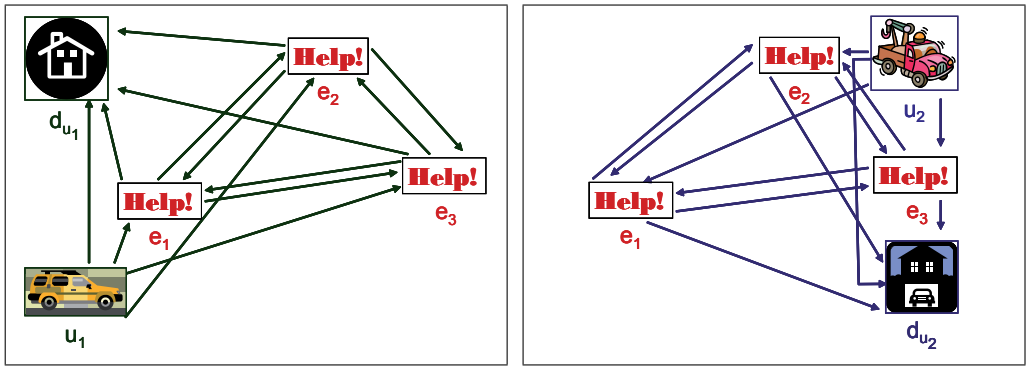
Die Stammdaten für die Einheiten bzw. die Aufträge lauten

	u_1	u_2
c_u^{drv}	5	10
c_u^{srv}	10	20
c^{ot}	15	15
t^{start}	5	0
t^{end}	240	120

	e_1	e_2	e_3	e_4
Θ_e^r	-10	0	-15	15
Θ_e^d	20	30	15	45
c^{late}	15	10	50	40
δ_e	30	30	40	5

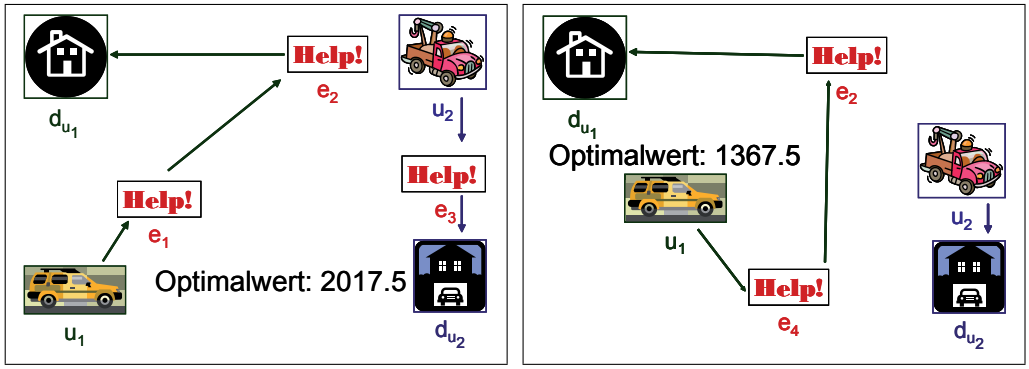
(1.58)

Um das erste Schnappschussproblem lösen zu können, müssen wir nun für jede Einheit einen Graphen konstruieren. In diesem Beispiel gehen wir davon aus, dass jede Einheit jeden Auftrag bedienen könnte. Den Graphen für Einheit u_1 zeigt Abbildung 1.2(a), den für u_2 1.2(b).



(a) Graph G_{u_1} (b) Graph G_{u_2}

Abbildung 1.2: Graph des ersten Schnappschussproblems



(a) Lösung des ersten Problems (b) Lösung des zweiten Problems

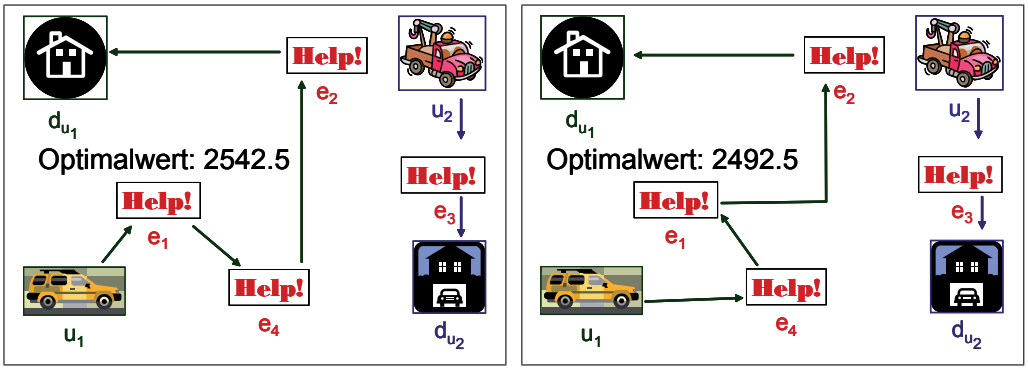
Abbildung 1.3: Lösung der Schnappschussprobleme aus Beispiel 1.5.1

Nun kann man das erste Fluss-basierte Modell in einen kommerziellen Löser eingeben und erhält folgende Variablenbelegung:

Variable	$y_{u_1}^{e_1,e_2}$	$y_{u_1}^{e_2,d_{u_1}}$	$y_{u_1}^{o_{u_1},e_1}$	$y_{u_2}^{e_3,d_{u_2}}$	$y_{u_2}^{o_{u_2},e_3}$	t_{u_1,e_2}^{l}	$t_{u_1}^{d_{u_1}}$	$t_{u_1}^{e_1}$	$t_{u_1}^{e_2}$	$t_{u_2}^{d_{u_2}}$	$t_{u_2}^{e_3}$
Wert	1	1	1	1	1	22	99.5	10	52	62.5	5

(1.59)

Alle anderen Variablen sind null. Die Kosten belaufen sich auf 2017.5. Diese Lösung entspricht der Tourenplanung in Abbildung 1.3(a). Das daraus resultierende gemischt ganzzahlige Problem hat in unserem Fall 44 Variablen und 53 Nebenbedingungen. Nun wird dieser Einsatzplan umgesetzt. Einige Zeit später wird dann Auftrag e_4 bekannt. Der alte Plan wird nun verworfen und ein neuer mit den veränderten Daten berechnet. Die Lösung zeigt Abbildung 1.3(b). Man beachte dabei, dass sich die Startpositionen der Einheiten u_1 und u_2 gegenüber dem ersten Schappschussproblem verändert haben, da die Einheiten einige Zeit nach



(a) Online Lösung (b) Offline Lösung

Abbildung 1.4: Gesamtlösung des Problems aus Beispiel 1.5.1

dem ersten Plan gearbeitet haben. Die Kosten des neuen Plans sind 1367.5.

Jetzt wollen wir natürlich wissen, wie hoch die Gesamtkosten des tatsächlich umgesetzten Einsatzplanes sind. Wir können nicht einfach die Kosten beider Lösungen addieren, da der erste Plan nicht bis zum Ende durchgeführt wurde und der zweite erst später einsetzt. Den umgesetzten Einsatzplan zeigt Abbildung 1.4(a). Wenn wir die Kosten dieser Lösung neu ausrechnen bekommen wir 2542.5.

Es stellt sich die Frage, was wir hätten sparen können, wenn wir die Zukunft gekannt hätten, also von e_4 im Voraus gewusst hätten (*experimentelle kompetitive Analyse*). Diese Offline-Lösung zeigt Abbildung 1.4(b). Ihre Kosten belaufen sich auf 2492.5. Die Online-Lösung aus der Reoptimierungspolitik ist also nur um 2% schlechter.

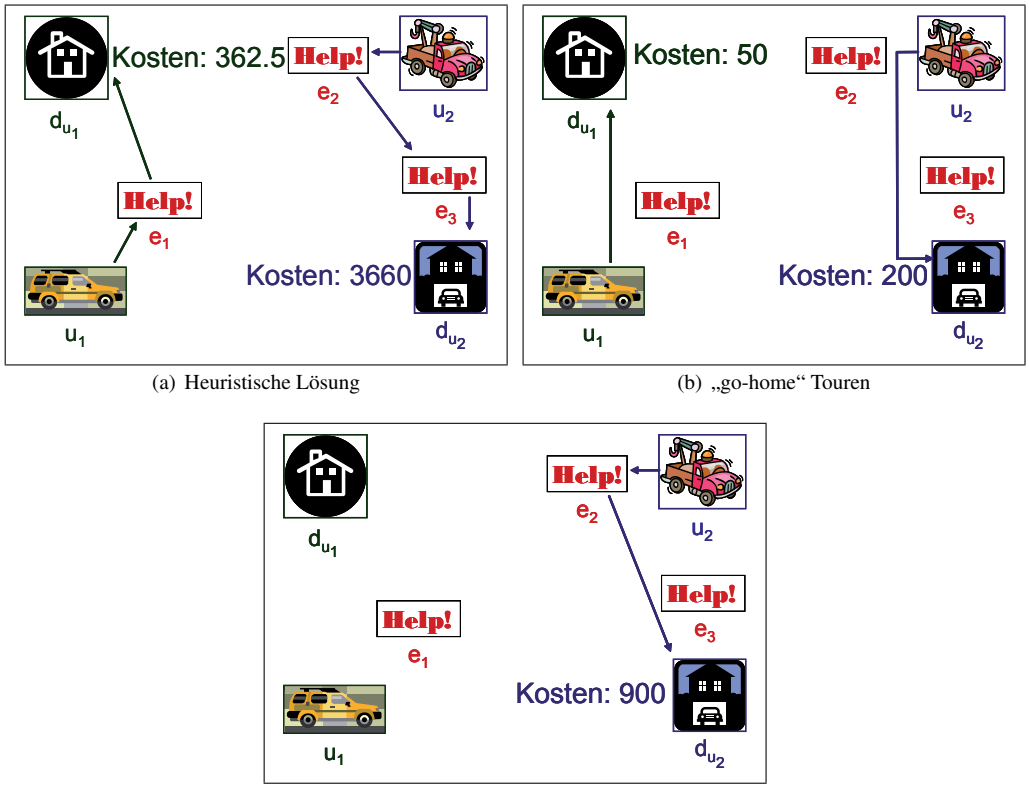
Wichtig für die effiziente Lösung des gemischt ganzzahligen Problems sind gute untere Schranken. Kommerzielle Löser ermitteln diese über die Lineare Relaxation, die man erhält, wenn man $x \in \{0,1\}$ durch $0 \leq x \leq 1$ ersetzt. Das resultierende Problem kann dann mit Hilfe des Simplexalgorithmus schnell gelöst werden. Wir geben nachfolgend für die drei Lösungen jeweils den Wert der optimalen gemischt ganzzahligen Lösung und den Wert der LP Relaxation an:

Problem	Optimalwert	LP Relaxation	Lücke
Schnappschussproblem 1	2017.5	1375.55	32%
Schnappschussproblem 2	1367.5	746.045	45%
Offline-Problem	2492.5	1493.8	40%

Die sogenannte *Ganzzahligkeitslücke* des Flussmodells ist also schon im kleinen Beispiel recht groß, was bei größeren Problemen dazu führt, dass der Lösungsprozess sehr lange dauert. Dies ist natürlich wegen des Echtzeitaspekts des ADAC-Problems sehr ungünstig.

Im Folgenden sehen wir, dass die Ganzzahligkeitslücke des Tourenmodells wesentlich kleiner ist.

Beispiel 1.5.2
Wir beginnen mit der heuristischen Lösung aus Abbildung 1.5(a). Weiterhin nehmen wir zu der Tourenmenge die „go-home“ Touren mit hinzu (Abbildung 1.5(b)). Für den Simplexalgorithmus brauchen wir noch eine weitere linear unabhängige Spalte, wir wählen die Tour aus Abbildung 1.5(c).



(a) Heuristische Lösung

(b) „go-home“ Touren

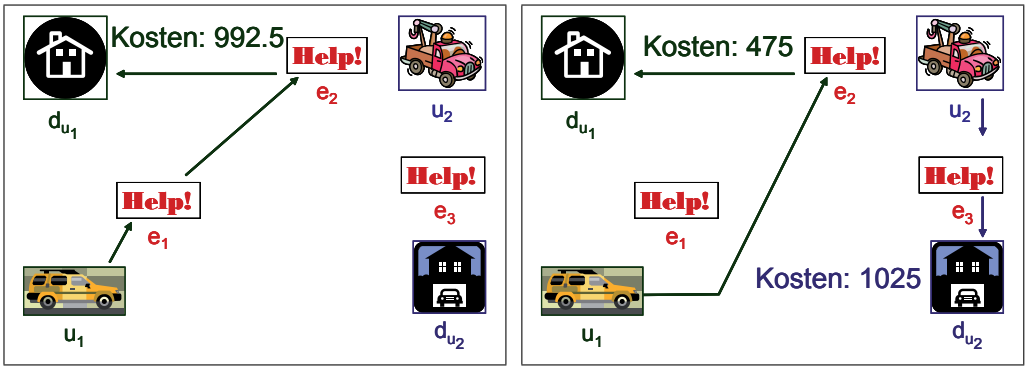
(c) Zusätzliche Tour für Basislösung

Abbildung 1.5: Startlösung für das Tourenmodell

Mit dieser Startkonfiguration lösen wir zuerst das LP und suchen nach Touren mit negativen reduzierten Kosten. Für Einheit u_1 werden wir dabei mit der Tour aus Abbildung 1.6(a) fündig. Wir nehmen diese Tour zu unserem reduzierten Problem hinzu und lösen das LP erneut. Mit den veränderten reduzierten Kosten finden wir die zwei Touren aus Abbildung 1.6(b).

Diese zwei Touren werden wieder hinzugefügt und das LP erneut gelöst. Für die neuen dualen Preise ergeben sich keine Touren mehr mit reduzierten Kosten. Wir haben also das nicht reduzierte Tourenmaster-Problem optimal gelöst. Das letzte reduzierte lineare Problem lautet:

$$\text{Minimiere } 362.5x_1 + 3660x_2 + 50x_3 + 200x_4 + 900x_5 + 992.5x_6 + 475x_7 + 1025x_8 \tag{1.60}$$



(a) Neue Tour für u_1 (b) Neue Touren für u_1 und u_2

Abbildung 1.6: Generierte Touren

unter den Nebenbedingungen $x_i \geq 0$, $i = 1, \ldots, 8$, und

$$\begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & 1 & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 \\ 1 & 0 & 1 & 0 & 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 1 & 1 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \\ x_7 \\ x_8 \end{pmatrix} = \begin{pmatrix} 1 \\ 1 \\ 1 \\ 1 \\ 1 \end{pmatrix} \tag{1.61}$$

Da die Optimallösung $x_6 = x_8 = 1$ bereits ganzzahlig ist, haben wir sogar das ganzzahlige Tourenmasterproblem 1.3.3 gelöst und die Lücke zwischen der LP-Relaxation und der ganzzahligen Lösung ist null.

Diese Verschärfung der unteren Schranke aus der LP-Relaxierung ist der Grund, warum nur über das Tourenmodell ein echtzeittaugliches Verfahren für die Lösung des Schnappschussproblems möglich ist. Theoretisch ist diese Verbesserung der LP-Relaxierung dadurch begründbar, dass das Tourenmodell durch eine Dantzig-Wolfe-Zerlegung aus dem Flussmodell entsteht (siehe [DDSS95] für eine Standard-Referenz zu diesem Aspekt für zeitbeschränkte Vehicle-Routing-Probleme und [Sch06] für eine ausführliche Ausarbeitung zum ADAC-Problem).

Was kam nun in der Praxis heraus?

Die Schnappschussprobleme aus Produktionsdaten des ADAC konnten mit dem oben skizzierten Tourenmodell und dem Spaltengenerierungs-Verfahren (plus ein paar Tricks ...) fast ausnahmslos mit kleinen Optimalitätslücken (5 %) gelöst werden, so dass unser Verfahren zur Lösung der Schnappschussprobleme als geeignet nachgewiesen ist [KRT02]. Genauso sieht es bei zufälligen Daten für das Aufzugsteuerungsproblem aus.

Zu bemerken ist, dass über das Flussmodell schon kleine Probleme mit 4 Einheiten und 8 Aufträgen z. B. durch ILOG cplex 10 nicht mehr in Sekunden, sondern nur in Minuten ge-

löst werden können, da die Ganzzahligkeitslücke des Flussmodells zu groß ist. Die Original-Schnappschussprobleme können auch in Stunden nicht mit dem Flussmodell gelöst werden.

Bleibt noch die Frage, ob die Reoptimierungspolitik vernünftig funktioniert. Auch das ließ sich in beiden Fällen positiv im Rahmen des Modells durch vergleichende Simulationsstudien beantworten [GKRT02, HKR06, FR06].

Ein Punkt aber bleibt offen beim ADAC-Problem: Außerhalb des Modells sind auch im Modell bekannte Daten unsicher (vor allem Fahrzeitschätzungen). Und bzgl. dieser Unsicherheiten sind die Ergebnisse beim ADAC nicht so robust, wie wir gerne hätten. Die mathematisch fundierten Verfahren der Robusten oder Stochastischen Optimierung sind aber leider noch nicht so weit, so große Probleme in Echtzeit zu lösen.

Beim Aufzugproblem ist die Datenunsicherheit (Fahrzeitschätzung) wegen der recht stabilen Födergeschwindigkeit der Fördertechnik weitaus geringer. Daher erwarten wir eine sehr gute Praxistauglichkeit unseres Verfahrens. Hier muss aber noch eine Anpassung des Modells an die real existierenden Fördertechnik-Layouts geschehen. Das ist in Arbeit.

Trotzdem sollte man Folgendes beim Modellieren nie vergessen:

Modellierungsprozess 1.5.3
Evaluiere die Ergebnisse des Modells und fange ggf. wieder von vorn an.

1.6 Zusammenfassung und Ausblick

Wir haben in diesem Beitrag gezeigt, mit welchen Schritten man dynamische diskrete Optimierungsprobleme modellieren kann. Manche dynamischen Optimierungsprobleme gestatten die sinnvolle Verwendung einer *Reoptimierungspolitik*. Diese führt die Lösung des dynamischen Problems mit unbekannten Informationen auf die sukzessive Lösung von statischen *Schnappschussproblemen* mit bekannten Informationen zurück, die dann in *Echtzeit* gelöst werden müssen.

Die Modellierung der Schnappschussprobleme kann durch geeignete *abstrakte Strukturen* (meist Graphen, hier: das Transportnetz) für aus Anwendersicht durchaus verschiedene Problemstellungen identisch durchgeführt werden. Damit können Lösungverfahren mit Hilfe der abstrakten Struktur als Schnittstelle so implementiert werden, dass sie für verschiedene Probleme benutzt werden können (hier: ADAC- und Aufzug-Problem).

Bei der echtzeittauglichen Lösung von Schnappschussproblemen aus der Ganzzahligen Linearen Optimierung ist es wichtig, dass die *Schranken* aus der LP-Relaxierung scharf genug sind. Dies wird i. d. R. eher mit Modellen erreicht, welche Variablen haben, die mit kompletten Teilkonfigurationen (hier: Touren einzelner Einheiten) indiziert sind, auch wenn dadurch astronomisch viele Variablen entstehen. Die Lösung dieser Modelle kann nämlich oft mit der Methode der *dynamischen Spaltengenerierung* geschehen, die ausnutzt, dass Optimierung i. d. R. schneller geht als Enumerierung.

Modellierung geschieht in unserem Wirkungsbereich in einem *Kreislauf* „Anwendung – Abstraktion – Modell – Algorithmen – Implementierung – Testen – Anwendung“ so lange, bis man sich zufrieden gibt. Anwendungsprobleme sind nie wirklich abschließend gelöst.

Herausforderungen der Zukunft sind die bessere Integration von *Robustheitsanforderungen* und *stochastischer Informationen*. Obwohl Modellierungskonzepte existieren, sind die Skalen

der angreifbaren Probleme dort noch wesentlich kleiner als für die reinen Reoptimierungspolitiken aus diesem Beitrag. Wir arbeiten daran.

Literaturverzeichnis

[DDSS95] DESROSIERS, JAQUES, YVAN DUMAS, MARIUS M. SOLOMON und FRANÇOIS SOUMIS: *Time Constraint Routing and Scheduling*. In: BALL, MICHAEL, TOM MAGNANTI, CLYDE MONMA und GEORGE NEWHAUSER (Herausgeber): *Network Routing*, Band 8 der Reihe *Handbooks in Operations Research and Management Science*, Kapitel 2, Seiten 35–140. Elsevier, Amsterdam, 1995.

[FR06] FRIESE, PHILIPP und JÖRG RAMBAU: *Online-optimization of a multi-elevator transport system with reoptimization algorithms based on set-partitioning models*. Discrete Applied Mathematics, 154:1908–1931, 2006.

[GKR01a] GRÖTSCHEL, MARTIN, SVEN O. KRUMKE und JÖRG RAMBAU: *Online Optimization of Complex Transportation Systems*. In: GRÖTSCHEL, MARTIN, SVEN O. KRUMKE und JÖRG RAMBAU (Herausgeber): *Online Optimization of Large Scale Systems*, Seiten 705–730. Springer, 2001.

[GKR+01b] GRÖTSCHEL, MARTIN, SVEN O. KRUMKE, JÖRG RAMBAU, THOMAS WINTER und UWE T. ZIMMERMANN: *Combinatorial Online Optimization in Real Time*. In: GRÖTSCHEL, MARTIN, SVEN O. KRUMKE und JÖRG RAMBAU (Herausgeber): *Online Optimization of Large Scale Systems*, Seiten 679–704. Springer, 2001.

[GKRT02] GRÖTSCHEL, MARTIN, SVEN O. KRUMKE, JÖRG RAMBAU und LUIS M. TORRES: *Online-dispatching of automobile service units*. In: LEOPOLD-WILDBURGER, ULRIKE, FARNZ RENDL und GERHARD WÄSCHER (Herausgeber): *Operations Research Proceedings*, Seiten 168–173. Springer, 2002.

[HKR00] HAUPTMEIER, DIETRICH, SVEN O. KRUMKE und JÖRG RAMBAU: *The online dial-a-ride problem under reasonable load*. In: *Proceedings of the 4th Italian Conference on Algorithms and Complexity*, Band 1767 der Reihe *Lecture Notes in Computer Science*, Seiten 137–149. Springer, 2000.

[HKR06] HILLER, BENJAMIN, SVEN O. KRUMKE und JÖRG RAMBAU: *Reoptimization Gaps versus Model Errors in Online-Dispatching of Service Units for ADAC*. Discrete Applied Mathematics, 154:1897–1907, 2006.

[KR02] KRUMKE, SVEN O. und JÖRG RAMBAU: *Probieren geht über Studieren? Entscheidungshilfen für kombinatorische Online-Optimierungsprobleme in der innerbetrieblichen Logistik*. at – Automatisierungstechnik, 50:568–575, 2002.

[KRT02] KRUMKE, SVEN O., JÖRG RAMBAU und LUIS MIGUEL TORRES: *Realtime-Dispatching of Guided and Unguided Automobile Service Units with Soft Time Windows*. In: MÖHRING, ROLF H. und RAJEEV RAMAN (Herausgeber): *Algorithms –*

ESA 2002, 10th Annual European Symposium, Rome, Italy, September 17–21, 2002, Proceedings, Band 2461 der Reihe *Lecture Notes in Computer Science*. Springer, 2002.

[Ram05] RAMBAU, JÖRG: *Deferment Control for Reoptimization – How to Find Fair Reoptimized Dispatches*. In: ALBERS, SUSANNE, ROLF H. MÖHRING, GEORG CH. PFLUG und RÜDIGER SCHULTZ (Herausgeber): *Algorithms for Optimization with Incomplete Information*, Nummer 05031 in *Dagstuhl Seminar Proceedings*. Internationales Begegnungs- und Forschungszentrum (IBFI), Schloss Dagstuhl, Germany, 2005. <http://drops.dagstuhl.de/opus/volltexte/2005/66> [date of citation: 2005-01-01].

[Sch06] SCHWARZ, CORNELIUS: *Subgradientenmethoden zur Dynamischen Spaltengenerierung für die echtzeitfähige Einsatzplanung von Pannenhilfefahrzeugen*. Diplomarbeit, Universität Bayreuth, Fakultät für Mathematik, Physik und Informatik, 2006.

Sachverzeichnis