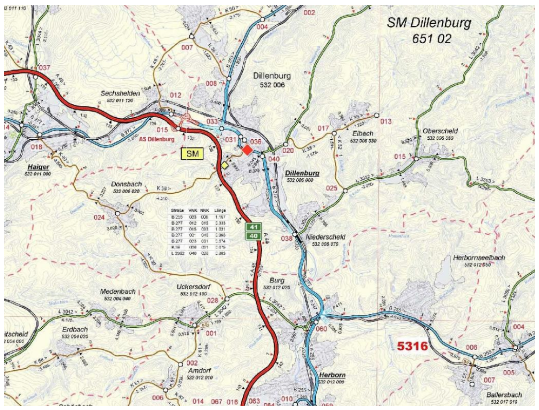
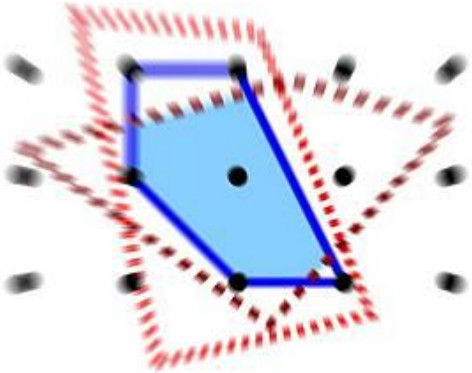
Ein Branch-and-Price-Algorithmus für den Straßenwinterdienst





Diplomarbeit
von
Sarah Kirchner

TU Darmstadt
Fachbereich Mathematik
Dezember 2010

Inhaltsverzeichnis

Abbildungsverzeichnis

Liste der Algorithmen

1 Einleitung

Der Straßenwinterdienstes soll die Leistungsfähigkeit des Straßennetzes und die Sicherheit des Straßenverkehrs gewährleisten. Seine Aufgabe besteht darin Verkehrsbehinderungen durch Schnee und Eis zu verhindern oder schnellstmöglich zu beseitigen.

Die Wirksamkeit des Straßenwinterdienstes hängt nicht zuletzt davon ab, wie schnell und wie umfassend das Straßennetz im Bedarfsfall geräumt werden kann. Denn je länger Schnee oder Eis die Straßen bedecken, desto größer ist die Beeinträchtigung und Gefährdung des Verkehrs. Um die schnelle und wirtschaftliche Räumung der Straßen zu gewährleisten, sollten die Fahrzeugrouten möglichst optimal geplant werden.

Es werden verschiedene Anforderungen an den Straßenwinterdienst gestellt. Diese Anforderungen sind unter anderem im Maßnahmenkatalog zur Verbesserung der Wirtschaftlichkeit des Straßenbetriebsdienstes [4] und im Leistungsheft für den Straßenbetriebsdienst (Leistungsbereich Winterdienst)[1] festgehalten. Dort ist festgelegt, dass die Fahrbahnspuren und Rampen der Bundesstraßen täglich zwischen 6.00 und 22.00 Uhr innerhalb von drei Stunden geräumt werden sollen. Schneit es nachts, soll der Räumeinsatz vor 6.00 Uhr beendet sein. Alle Fahrbahnspuren einer Richtung sollen gleichzeitig gestreut werden, um Gefahren beim Spurwechsel zu verhindern. Die Einsatzpläne sollen so optimiert werden, dass die Leerfahrten (Fahrten in denen das Fahrzeug nicht räumt oder streut) auf ein Mindestmaß reduziert werden. Die Räum- und Streueinsätze sollen an einer Straßenmeisterei beginnen und an der selben Meisterei auch enden. Die Tour die ein Fahrzeug in einem Einsatz fährt besteht aus der Räum- und Streustrecke, den Leerfahrten sowie den Nachladewegen. Der Schätzwert für die Geschwindigkeit liegt bei 35 km/h für Streueinsätze und bei 30 km/h für Räumeinsätze. Diese Werte sind als Durchschnittsgeschwindigkeiten für den Gesamteinsatz einschließlich aller Leerwege zu sehen.

Die Zuständigkeit für die Räumung der Straßen eines bestimmte Gebiets kann bei unterschiedlichen Trägern liegen. In Deutschland ist der Baulastträger einer Straße für deren Unterhaltung zuständig. So ist das Bundeslands zum Beispiel für die Räumung der Landesstraßen nicht aber der Autobahnen zuständig.

In dieser Arbeit wird nur der Räumeinsatz der Fahrzeuge betrachtet, nicht dagegen der Streueinsatz. Die Kapazitäten der Fahrzeuge für Streugut werden vernachlässigt und es werden keine Nachladestationen für Streugut berücksichtigt. Es wird davon ausgegangen, dass die Fahrzeuge mit einer Geschwindigkeit von 30 km/h fahren. Die Touren sollen so geplant werden, dass innerhalb von drei Stunden alle Straßen im Zuständigkeitsbereich geräumt sind und dass jedes Fahrzeug an sein Ursprungsdepot zurückgekehrt ist. Außerdem wird davon ausgegangen, dass die Anzahl der zur Verfügung stehenden Fahrzeuge nicht beschränkt ist. Dieses Problem kann als *„Capacitated-Arc-Routing-Problem (CARP)“* mit mehreren Depots formuliert werden, wie in Kapitel 6 erläutert wird.

Ziel dieser Arbeit ist die Entwicklung eines Branch-and-Price-Verfahrens für den Straßenwinterdienst. Dazu wird das Problem in mehrere Teilprobleme dekomponiert, die Subprobleme und das Masterproblem. In den Subproblemen wird versucht Fahrzeugrouten zu finden, die an einem der Depots beginnen und enden und die die maximal zulässige Zeit nicht überschreiten. Im Masterproblem wird versucht, aus den Touren, die in den Subproblemen gefunden wurden, eine optimale Kombination zu finden.
Wenn möglich, soll das Straßenwinterdienstproblem für den Zuständigkeitsbereich der Straßenmeistereien des Amts für Straßen- und Verkehrswesen Dillenburg unter Verwendung des Branch-and-Price-Verfahrens gelöst werden. Es werden dazu praktische Daten des Straßennetzes in den Kreisen Lahn-Dill und Limburg-Weilburg verwendet. Informationen zum Straßennetz findet man bei Hessische Straßen- und Verkehrsverwaltung [12].

1.1 Literatur

Die meisten Lösungsverfahren für das Straßenwinterdienstproblem sind Heuristiken. Eglese [10] hat eine Heuristik für die Routenplanung im Straßenwinterdienst mit mehreren Depots entwickelt. Zuerst wird das Chinese-Postman-Problem (finden einer Tour durch das Netzwerk, die alle Kante enthält) für das Straßennetzwerk gelöst. Für die Leerfahrten in der Lösung des Chinese-Postman-Problems werden Kanten zum Graphen hinzugefügt. Aus diesem erweiterten Graphen wird dann eine Menge von Kreisen gebildet. Diese Kreise werden anschließend zu Touren zusammengefügt, die an einem Depot beginnen und enden. Perrier et al. [25] haben eine Formulierung des Problems vorgestellt, in der Prioritäten zwischen den verschiedenen Straßen, verschiedene Räum- und Durchfahrgeschwindigkeiten modelliert werden können. Außerdem wurden in diesem Artikel zwei Heuristiken zur Lösung dieses Problems vorgestellt. Allerdings wird nur ein Depot betrachtet, da das Straßennetzwerk häufig schon in Distrikte aufgeteilt ist. Weitere Heuristiken für das Straßenwinterdienstproblem können in Perrier et al. [24] nachgelesen werden.

Für das CARP wurden auch mehrere exakte Lösungsverfahren vorgestellt. Longo et al. [19] schlagen zur Lösung des CARP eine Transformation in das knotenbasierte Vehicle-Routing-Problem vor, für das es schon viele sehr gute exakte Lösungsverfahren gibt. Ein Branch-and-Price-Verfahren zur Lösung des CARP ohne vorherige Transformation in ein äquivalentes knotenbasiertes Routing-Problem wird in Letchford und Oukil [18] vorgestellt.

1.2 Aufbau der Arbeit

In Kapitel 2 werden die notwendigen Grundlagen kurz vorgestellt und die verwendeten Notationen erläutert. Kapitel 3 gibt eine Einführung in das Branch-and-Price-Verfahren. Anhand eines ganzzahligen Programms wird erläutert, wie das Programm in Master- und Subprobleme dekomponiert werden kann und wie das Masterproblem mit Hilfe von Spaltenerzeugung und dem Branch-and-Bound-Verfahren gelöst werden kann. In Kapitel 4 wird das Branch-and-Price-Verfahren für den in Routing-Problemen sehr häufig autretenden

Fall eines Set-Partitioning-Masterproblems erläutert. Es wird insbesondere auf die speziell in diesem Fall anwendbare Branchingregel von Ryan und Foster eingegangen. Kapitel 5 erläutert das Ressource-Constrained-Shortest-Path-Problem. In vielen Routing-Problemen und auch im Straßenwinterdienstproblem sind bei Anwendung eines Branch-and-Price-Verfahrens die Subprobleme Ressource-Constrained-Shortest-Path-Probleme. Es wird eine kurze Einfürung in die Lösung des Problems mittels dynamischer Programmierung gegeben. In Kapitel 6 wird das Straßenwinterdienstproblem mathematisch als CARP formuliert und der dem Problem unterliegende Graph beschrieben. Danach wird in Kapitel 7 erläutert, wie das Branch-and-Price Verfahren auf das Straßenwinterdienstproblem angewendet werden kann. In Kapitel 8 werden die Ergebnisse des Verfahrens für einen kleinen Testgraph vorgestellt.
Da das Problem für das Straßennetz im Zuständigkeitsbereich der Straßenmeisterein im Landkreis Lahn-Dill und Limburg-Weilburg zum Zeitpunkt der Fertigstellung der Arbeit nicht gelöst werden konnte, wird in Kapitel 9 das Ergebnis einer Heuristik für das Problem gegeben. Es werden außerdem zwei Veränderungen für das Verfahren vorgeschlagen, mit denen sich möglicherweise bessere Ergebnisse erzielen lassen.

2 Grundlagen

In diesem Kapitel werden die in dieser Arbeit verwendeten Notationen und Sätze vorgestellt. Es handelt sich dabei vor allem um Grundlagen der Optimierung und Graphentheorie.

2.1 Optimierung

Definition 2.1 (Polyeder)
Sei $A \in \mathbb{R}^{m\times n}$ und $b \in \mathbb{R}^m$.Eine Menge $P \subseteq \mathbb{R}^n$ mit

$$P = P(A, b) = \{x \in \mathbb{R}^n : Ax \leq b\},$$

heißt **Polyeder**. *Ein beschränktes Polyeder heißt* **Polytop**.

Definition 2.2 (Lineares Programm (LP))
Sei $c \in \mathbb{R}^n$, $A \in \mathbb{R}^{m\times n}$, $b \in \mathbb{R}^m$, dann heißt

$$\begin{aligned} \min\, & c^T x \\ & Ax \leq b \\ & x \geq 0 \\ & x \in \mathbb{R}^n \end{aligned}$$

lineares Programm (LP)

Die zulässige Menge eines LPs ist durch das Poyeder $P = P(A, b)$ gegeben. Ein LP hat daher folgende Form

$$\begin{aligned} \min\, & c^T x \quad \text{s.t.} \\ & x \in P(A, b) \end{aligned}$$

Definition 2.3 (Konvexkombination)
Ein Vektor $x \in \mathbb{R}^n$ heißt **Konvexkombination** *von Vektoren $x_1, \ldots, x_m \in \mathbb{R}^n$, falls es Skalare λ_i mit*

$$\lambda_i \geq 0 \quad \textit{und} \quad \sum_{i=1}^{m} \lambda_i = 1$$

gibt, so dass

$$x = \sum_{i=1}^{m} \lambda_i x_i$$

Ein Vektor $x \in \mathbb{R}^n$ ist eine Konvexkombination von Vektoren der Menge $X \subset \mathbb{R}^n$, wenn es eine Zahl $m \in \mathbb{N}$, sowie Vektoren $x_1, \ldots, x_m \in X$ gibt, so dass x eine Konvexkombination der Vektoren $x_1, \ldots, x_m$ ist.

Definition 2.4 (Konvexe Hülle)
Gegeben sei eine Menge $X \subset \mathbb{R}^n$. Die **konvexe Hülle** *der Menge X ist die Menge, die alle Konvexkombinationen der Elemente aus X enthält. Sie wird mit $conv(X)$ bezeichnet.*

Definition 2.5 (Konische Kombination)
Ein Vektor $x \in \mathbb{R}^n$ ist eine **konische Kombination** *der Vektoren $x_1, \ldots, x_m \in \mathbb{R}^n$, falls es Skalare λ_i gibt mit $\lambda_i \geq 0$, so dass*

$$x = \sum_{i=1}^{m} \lambda_i x_i$$

Ein Vektor $x \in \mathbb{R}^n$ ist eine konische Kombination von Vektoren der Menge $X \subset \mathbb{R}^n$, wenn es eine Zahl $m \in \mathbb{N}$ sowie Vektoren $x_1, \ldots, x_m \in X$ gibt, so dass x eine konische Kombination der Vektoren $x_1, \ldots, x_m$ ist.

Definition 2.6 (Konische Hülle)
Gegeben sei eine Menge $X \subset \mathbb{R}^n$. Die **konische Hülle** *der Menge X ist die Menge, die alle konischen Kombinationen der Elemente aus X enthält. Sie wird mit $cone(X)$ bezeichnet.*

Satz 2.1 (Minkowski-Weyl)
Eine Menge $P \subset \mathbb{R}^n$ ist genau dann ein Polyeder, wenn es endliche Mengen $V, E \in \mathbb{R}^n$ gibt, so dass

$$P = conv(V) + cone(E)$$

Die Elemente der Menge V heißen Extrempunkte *von P. Die Elemente der Menge E heißen* Extremstrahlen *von P. Beweis siehe Schrijver [28].*

Definition 2.7 (Ganzzahliges Programm (IP))
Sei $c \in \mathbb{R}^n$, $A \in \mathbb{R}^{m \times n}$, $b \in \mathbb{R}^m$, dann heißt

$$\begin{aligned} \min\, & c^T x \\ & Ax \leq b \\ & x \in \mathbb{Z}^n \end{aligned} \tag{2.1}$$

ganzzahliges Programm (IP)

Ein IP ist kein lineares Programm, da der Zulässigkeitsbereich

$$S = \{x \in P(A, b) : x \in \mathbb{Z}^n\}$$

eine diskrete Menge und im Allgemeinen kein Polyeder ist.

Ist S endlich, dann ist nach Satz 2.1 $conv(S)$ ein Polytop. Ein IP ist dann äquivalent zu dem linearen Programm

$$\begin{aligned} &\min c^T x \quad \text{s.t.} \\ &x \in conv(S). \end{aligned}$$

Definition 2.8 (LP-Relaxierung)
Wird in einem IP (2.1) *die Ganzzahligkeitsbedingung weggelassen, erhält man die LP-Relaxierung des IPs*

$$\begin{aligned} \min\, & c^T x \\ & Ax \leq b \end{aligned}$$

Satz 2.2
Gegeben sei ein Poyeder $P = P(A, b)$ mit $A \in \mathbb{Z}^{m \times n}$ und $b \in \mathbb{Z}^m$ und zwei endliche Mengen $V = \{v_i\}_{l \in L}$ und $E = \{e_j\}_{j \in J}$, mit $P = conv(V) + cone(E)$. S bezeichne die Menge der ganzzahligen Punkte in P, d.h. $S = \{x \in P : x \in \mathbb{Z}^n\}$. Dann gibt es eine endliche Menge von Punkten $\{x_i\}_{i \in I}$ in S, sodass gilt:

$$S = \left\{ x : x = \sum_{i \in I} \lambda_i x_i + \sum_{j \in J} \beta_j e_j, \sum_{i \in I} \lambda_i = 1, \lambda_i \in \mathbb{Z}_+ \text{ für } i \in I, \beta_j \in \mathbb{Z}_+ \text{ für } j \in J \right\}$$

Außerdem ist $conv(S)$ ein rationales Polyeder. Für den Beweis siehe Nemhauser und Wolsey [23, Kapitel I.4.6])

Definition 2.9 (Set-Partitioning-Problem)
Gegeben sei eine endliche Grundmenge $E = \{1, \ldots, m\}$ und eine Liste von Teilmengen $F_j \subset E$ $(j = 1, \ldots, n)$ von Teilmengen von E. Mit jeder Teilmenge sind Kosten c_j assoziiert. Das Problem eine bzgl. c minimale Auswahl von Teilmengen zu finden, so dass jedes Element der Grundmenge E in genau einer ausgewählten Teilmenge vorkommt, heißt **Set-Partitioning-Problem**. *Es kann als 0-1-Programm formuliert werden.*
Variablen des Programms:

$$x_j = \begin{cases} 1 & \text{falls Teilmenge } F_j \text{ gewählt wird} \\ 0 & \text{sonst} \end{cases}$$

Die Nebenbedingungsmatrix ist eine 0-1-Matrix $A \in \{0,1\}^{m \times n}$ wobei

$$a_{ij} = \begin{cases} 1 & \text{falls Teilmenge } F_j \text{ Element } i \in E \text{ enthält} \\ 0 & \text{sonst} \end{cases}$$

Das 0-1-Programm lautet:

$$\begin{aligned} \min\, & c^T x \quad s.t. \\ & Ax = \mathbb{1} \\ & x \in \{0,1\}^n \end{aligned}$$

Satz 2.3 (Farkas Lemma)
Entweder gilt

$$\left\{x \in \mathbb{R}_+^n : Ax = b, x \geq 0\right\} \neq \emptyset$$

oder es gibt einen Vektor $\pi \in \mathbb{R}_+^m$*, so dass*

$$\pi A \geq 0 \quad \textit{und} \quad \pi b > 0$$

Für den Beweis siehe z.B. Schrijver [28].

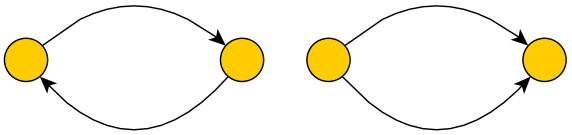
2.2 Graphentheorie

Definition 2.10 (Gerichteter Graph)
Ein **gerichteter Graph** $G = G(V, A, g)$ *ist ein geordnetes Tripel, wobei* V *und* A *disjunkte Mengen sind und* $g : A \to V^2$ *eine Abbildung ist. Die Elemente der Menge* V *heißen Knoten und die Elemente der Menge* A *heißen Kanten des Graphen* G*. Die Abbildung* $g : A \to V^2$ *ordnet jeder Kante ihren Anfangs- und Endknoten zu,*

$$g(a) = (v, w).$$

Man sagt dann Kante a *ist von* v *nach* w *gerichtet.*

Definition 2.11 (Mehrfachkanten, parallele Kanten)
Ein gerichteter Graph $G(V, A, g)$ *kann zwischen zwei Knoten mehrere Kanten haben, diese nennt man* **Mehrfachkanten**. *Haben die Mehrfachkanten in einem gerichteten Graphen die selbe Richtung, so nennt man die Mehrfachkanten* **parallel**.



(d) Mehrfachkanten (e) Parallele Kanten

Abbildung 2.1: Mehrfachkanten, parallele Kanten

Definition 2.12 (Pfad, Tour)
Sei $G(V, A, g)$ *ein gerichteter Graph. Eine Folge*

$$(v_0, a_0, v_1, a_1, \ldots, a_{p-1}, v_p)$$

von abwechselnd Knoten und Kanten für die gilt

$$g(a_i) = (v_i, v_{i+1}), \ \textit{für } i < p$$

wird **Pfad** *genannt. Eine Pfad heißt* einfach, *falls jede Kante höchstens einmal in diesem Pfad vorkommt. Ein Pfad wird* **Tour** *genannt falls Anfangs- und Endknoten übereinstimmen. Eine Tour ist ein Pfad mit* $v_0 = v_p$*.*

Definition 2.13 (Knotengrad)
Sei $G(V, A, g)$ ein gerichteter Graph. Die Anzahl der Kanten aus G, deren Endknoten v ist, wird **Eingangsgrad** *des Knotens v genannt und mit δ_v^- bezeichnet. Die Anzahl der Kanten, die v als Anfangsknoten haben, wird* **Ausgangsgrad** *des Knotens v genannt und mit δ_v^+ bezeichnet. $\delta^-(v) \subset A$ bezeichnet die Menge der Kanten, deren Endknoten v ist und $\delta^+(v) \subset A$ bezeichnet die Menge der Kanten, deren Anfangsknoten v ist.*
Für eine Teilmenge W von V bezeichnet $\delta^+(W)$ die Menge aller Kanten, die ihren Anfangsknoten in W haben aber ihren Endknoten in $V \setminus W$.

3 Branch-and-Price für Ganzzahlige Programme

Das Branch-and-Price-Verfahren ist ein vielversprechendes Lösungsverfahren für große ganzzahlige Programme (IP), das die Methoden der Spaltenerzeugung mit dem Branch-and-Bound-Verfahren kombiniert.
Für viele Probleme der kombinatorischen Optimierung gibt es stark „ausgeweitete“ (engl. extended) Formulierungen mit sehr vielen Variablen. Ein typisches Beispiel dafür sind Tourenplanungsprobleme, wie das Straßenwinterdienstproblem. Die Anzahl der Variablen ist dabei häufig so groß, dass die Variablen nicht alle effizient explizit betrachtet werden können. In diesem Fall wird das Verfahren der Spaltenerzeugung eingesetzt, um die LP-Relaxierung des Problems zu lösen. Da die Lösung der LP-Relaxierung möglicherweise die Ganzzahligkeitsbedingungen des Problems verletzt, wird das Branch-and-Bound-Verfahren verwendet um eine ganzzahlige Lösung zu finden. Die Idee der Spaltenerzeugung besteht darin, ausgehend von einer kleinen Teilmenge der Variablen, eine Lösung für das Problem zu finden. Das auf diese Variablen eingeschränkte Problem kann unter Verwendung des Simplex-Algorithmus direkt gelöst werden. Anschließend wir überprüft, ob die Lösung des eingeschränkten Problems optimal für die LP-Relaxierung des Ausgangsproblems ist. Dies geschieht durch die Lösung des sogenannten *Pricing-Problems*. Hierbei wird überprüft, ob der Zielfunktionswert des Problems durch Hinzufügen von Variablen, die noch nicht im eingeschränkten Problem berücksichtigt werden, verbessert werden kann. Falls im Pricing-Problem eine solche Variable gefunden wurde, wird diese zum eingeschränkten Problem hinzugefügt und dieses reoptimiert. Anschließend muss wieder das Pricing-Problem gelöst werden. Wird im Pricing-Problem keine Variable mit negativen reduzierten Kosten gefunden, so ist die Lösung des eingeschränkten Problems optimal für die LP-Relaxierung des Ausgangsproblems.

Erfüllt diese Lösung nicht die Ganzzahligkeitsbedingung des Ausgangsproblems, so wird das Problem verzweigt. Das heißt die zulässige Menge des Problems wird so aufgeteilt, dass die aktuelle fraktionale Lösung ausgeschlossen wird. Es muss jedoch sichergestellt sein, dass alle zulässigen ganzzahligen Lösungen in einer der Teilmengen enthalten sind.

Um die LP-Relaxierungen der so neu entstandenen Probleme zu lösen, kann dann wieder Spaltenerzeugung verwendet werden. Wendet man Spaltenerzeugung innerhalb von Branch-and-Bound an, kann es aber auch zu Schwierigkeiten kommen (s. Johnson [15]). Bespielsweise sind Standardbranchingregeln möglicherweise nicht effizient anwendbar, weil sie die Struktur des Pricing-Problems zerstören.

Viele Probleme in der kombinatorischen Optimierung haben eine Struktur, in der man Blöcke von Variablen und Nebenbedingungen sowie Nebenbedingungen durch die diese Blöcke miteinander in Verbindung gebracht werden identifizieren kann. Die Idee der Dekomposition besteht darin, die verbindenden Nebenbedingungen auf einer übergeordneten Ebene im *Masterproblem* zu betrachten. Die anderen Blöcke werden unabhängig voneinander

ander in Teilproblemen behandelt. Probleme mit einer solchen Struktur lassen sich so reformulieren, dass sie sich für die Lösung durch das Branch-and-Price-Verfahren eignen. Eine ausführliche Einführung in das Branch-and-Price-Verfahren mit vielen Anwendungsbeispielen geben Wolsey [30] und Barnhart et al. [2] und Lübbecke und Desrosiers [17].

In diesem Kapitel wird besprochen, wie ein für Branch-and-Price geeignetes Problem reformuliert und dann durch das Branch-and-Price-Verfahren gelöst werden kann.

3.1 Probleme und ihre Reformulierung

Gegeben sei das ganzzahlige Programm

$$\begin{array}{rrrrl} & & & \min \sum_{k=1}^{K} c^k x^k & \text{s.t.} \\ A^1 x^1 & + \ldots & + A^K x^K & = & b \\ D^1 x^1 & & & \leq & d_1 \\ & \ddots & & & \vdots \\ & & D^K x^K & \leq & d_K \\ \text{wobei} & x^1 \in \mathbb{Z}^{n_1}, \ldots, x^K \in \mathbb{Z}^{n_K}. & & & \end{array} \tag{IP}$$

Dieses IP wird auch Originalproblem genannt. Die Nebenbedingungsmatrix von IP hat die in Abb. 3.1 dargestellte eingeschränkte Blockdiagonalform, die für jedes k einen Block D^k enthält, sowie die „Coupling Constraints“.

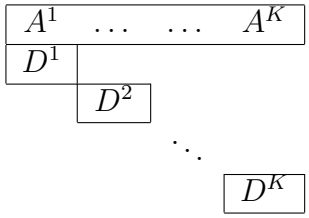


Abbildung 3.1: Struktur der Nebenbedingungsmatrix

Diese Struktur der Nebenbedingungsmatrix kann durch Dekomposition des Problems ausgenutzt werden. Die Dekomposition führt zu einer Formulierung des Problems mit sehr vielen Spalten, die daher häufig am besten durch Spaltenerzeugung gelöst werden kann. Der Dekompositionsansatz wurde erstmals von Dantzig und Wolfe [6] für Lineare Programme vorgeschlagen. Er basiert auf der Reformulierung der Polyeder

$$P_k = \left\{ x^k \in \mathbb{R}^{n_k} : D^k x^k \leq d_k \right\}$$

mit Hilfe von Satz 2.1. Der Dekompositionsansatz kann auf ganzzahlige Probleme übertragen werden. Man betrachtet statt der Polyeder P_k die diskrete Menge

$$P_k^* = \left\{ x^k \in P_k : x^k \in \mathbb{Z}^{n_k} \right\}$$

Es gibt zwei Möglichkeiten der Reformulierung, die Konvexifizierung und die Diskretisierung (siehe z.B. Vanderbeck [29]). Bei der Konvexifizierung wird nicht die Menge P_k^* reformuliert, sondern die konvexe Hülle $conv(P_k^*)$. Die hier verwendete Methode der Diskretisierung beruht auf einer Reformulierung der Menge P_k^*. Sind alle Variablen x_k der IPs binär, führen Konvexifizierung und Diskretisierung zu äquivalenten Masterproblemen.

Im Folgenden soll das oben vorgestellte IP reformuliert werden. Das IP kann notiert werden als

$$\begin{aligned} \min \sum_{k=1}^{K} c^k x^k \quad & s.t \\ & \sum_{k=1}^{K} A^k x^k = b \\ & x^k \in P_k^* \qquad k = 1, \ldots, K \end{aligned} \tag{3.1}$$

Die Menge P_k^* soll nun durch eine endliche Menge von Vektoren(Spalten) dargestellt werden. Ist der Polyeder P_k beschränkt, so ist die Menge P_k^* endlich. Sei also $P_k^* = \left\{ x^{k,t} \right\}_{t=1}^{T_k}$. Nach Satz 2.2 gilt

$$P_k^* = \left\{ x : x = \sum_{t=1}^{T_k} \lambda_t^k x^{k,t}, \sum_{t=1}^{T_k} \lambda_t^k = 1, \lambda_t^k \in \mathbb{Z}_+ \text{ für } t = 1, \ldots, T_k \right\} \tag{3.2}$$

Für jeden Punkt $x^k \in P_k^*$ gibt es demnach eine Darstellung

$$x^k = \sum_{t=1}^{T_k} \lambda_t^k x^{k,t} \tag{3.3}$$

wobei die Konvexitätsbedingung

$$\sum_{t=1}^{T_k} \lambda_t^k = 1 \quad \text{mit } \lambda_t^k \in \{0, 1\} \quad t = 1, \ldots, T_k$$

gelten muss.

Geht man davon aus, dass jedes Polyeder P_k beschränkt ist, erhält man durch Substitution der x^k in (3.1) durch (3.3) folgende Form für das IP

$$\begin{aligned} \min \sum_{k=1}^{K} \sum_{t=1}^{T_k} \lambda_t^k c^k x^{k,t} \quad & s.t \\ \sum_{k=1}^{K} \sum_{t=1}^{T_k} \lambda_t^k A^k x^{k,t} = b & \\ \sum_{t=1}^{T_k} \lambda_t^k = 1 \quad & k = 1, \ldots, K \\ \lambda_t^k \in \{0,1\} \quad & k = 1, \ldots, K \quad t = 1, \ldots, T_k \end{aligned} \tag{MP}$$

Diese Form des Problems wird im folgenden Masterproblem genannt und ist auch wieder ein IP. Der Unterschied zwischen (3.1) und dem Masterproblem (MP) besteht darin, dass die Mengen P_k^* durch endlich viele Punkte dagestellt werden.
Für jedes $x \in P_k^*$ gibt es eine Spalte $\begin{pmatrix} A^k x \\ e_k \end{pmatrix}$ in MP und eine Variable λ, mit Kosten $c^k x$.

Aus einer optimalen Lösung der LP-Relaxierung des Masterproblems $\widetilde{\lambda}$ kann man die Werte der Variablen des Originalproblems $\widetilde{x}^k$ berechnen.

$$\widetilde{x}^k = \sum_{t=1}^{T_k} \widetilde{\lambda}_{k,t} x^{k,t}$$

Da man eine ganzzahlige Lösung für das Originalproblem finden will, kann man auf diesen Variablen verzweigen. Näheres dazu in Abschnitt 3.4.

Im folgenden Abschnitt soll nun erläutert werden, wie die LP-Relaxierung des Masterproblems MP mithilfe von Spaltenerzeugung gelöst werden kann.

3.2 Lösen der LP-Relaxierung des Masterproblems

In einem Branch-and-Price-Verfahren wird die Methode der Spaltenerzeugung verwendet, um die LP-Relaxierung des Masterproblems (LMP) zu lösen, wenn die explizite Betrachtung aller Spalten das Masterproblem wegen ihrer Vielzahl nicht effizient ist. Seien

$c^{k,t} = c^k x^{k,t}$ und $A^{k,t} = A^k x^{k,t}$ für $t = 1, \ldots, T_k$. Für das LMP erhält man

$$\min \sum_{k=1}^{K} \sum_{t=1}^{T_k} \lambda_t^k c^{k,t} \quad s.t \tag{3.4}$$

$$\sum_{k=1}^{K} \sum_{t=1}^{T_k} \lambda_t^k A^{k,t} = b \tag{3.5}$$

$$\sum_{t=1}^{T_k} \lambda_t^k = 1 \quad k = 1, \ldots, K \tag{3.6}$$

$$\lambda_t^k \geq 0 \quad k = 1, \ldots, K \quad t = 1, \ldots, T_k \tag{3.7}$$

Im Folgenden seien π_i $(i = 1, \ldots, m)$ die zu den Nebenbedingungen (3.5) des (LMP) gehörenden dualen Variablen und μ_k $(k = 1, \ldots, K)$ die zu den Konvexitätsbedingungen (3.6) gehörenden dualen Variablen. Die Idee besteht nun darin, das LMP mit dem Simplex-Algorithmus zu lösen, wobei der Schritt verändert wird, in dem die Auswahl der Spalte stattfindet, die zur Basis hinzugefügt wird (s. Simplex-Algorithmus z.B in [28]). Der Grund dafür ist, dass das LMP in den meisten Fällen sehr viele Spalten hat, die möglicherweise nicht alle explizit bekannt sind. Statt also für jede Spalte des LMP die reduzierten Kosten zu berechnen, löst man K weitere Optimierungsprobleme, um eine (oder mehrere) Spalten mit negativen reduzierten Kosten zu finden.

Um das LMP mittels Spaltenerzeugung zu lösen, muss man schon eine Teilmenge der Spalten des LMP kennen oder das in Abschnitt 3.5 erläuterte Farkas-Pricing verwenden. Soll die Splatenerzeugung mit einer bereits bekannten Teilmenge der Spalten gestartet werden, muss für jedes k mindestens eine Spalte bekannt sein, sodass man eine zulässige Lösung für das LMP erhält. Es gibt verschiedene Möglichkeiten eine solche Startlösung zu finden. Unter anderem kann man solche Spalten durch eine heuristische Lösung des Originalproblems erhalten. Das auf diese Spalten eingeschränkte Problem wird *Restricted Master Problem (RMP)* genannt.

$$\begin{aligned} \min \widetilde{c}\widetilde{\lambda} \quad & s.t. \\ \widetilde{A}\,\widetilde{\lambda} &= \widetilde{b} \\ \widetilde{\lambda} &\geq 0 \end{aligned} \tag{RMP}$$

wobei $\widetilde{A}$ die Matrix der bekannten Spalten und damit eine Untermatrix der eigentlichen Nebenbedingungsmatrix des LMP ist. Die rechte Seite des LMP ist $\widetilde{b} = (b, \mathbb{1})$. Der Vektoren $\widetilde{c}$ und $\widetilde{\lambda}$ sind die mit einer Spalte assoziierten Kosten und Variablen.

Das RMP kann direkt durch Anwendung des Simplexverfahres gelöst werden. Sei $\widetilde{\lambda}^*$ eine optimale Lösung für das RMP und eine dual optimale Lösung (π, μ). Jede zulässige Lösung des RMP ist auch für das LMP zulässig. Daher ist der Zielfunktionswert $\widetilde{c}\widetilde{\lambda}^*$ des RMP eine

obere Schranke für den Zielfunktionswert des LMP.

Um festzustellen, ob die optimale Lösung des RMP auch optimal für das LMP ist, muss für jede Spalte des LMP überprüft weden, ob die reduzierten Kosten dieser Spalte negativ sind. Das heißt für alle k und alle $x \in X_k^*$ muss überprüft werden, ob für die Spalte $\begin{pmatrix} A^k x \\ e_k \end{pmatrix}$ des LMP

$$c^k x - \begin{pmatrix} \pi & \mu \end{pmatrix} \begin{pmatrix} A^k x \\ e_k \end{pmatrix} \leq 0$$

gilt.

Oft ist es ineffizient, diese Ungleichung für jeden Punkt $x \in X_K^*$(bzw. jede Spalte des LMP) zu überprüfen, . Man löst stattdessen für jedes k das Optimierungsproblem

$$\begin{aligned} \min c^k x^k - \pi A^k x^k - \mu_k \quad \text{s.t.} \\ x^k \in X_k^* \end{aligned} \tag{3.8}$$

Dieses Problem wird Pricing-Problem genannt.

Wenn in keinem dieser k Optimierungsprobleme der Zielfunktionswert echt kleiner null ist, dann ist (π, μ) dual zulässig für LMP. Da $\widetilde{\lambda}^*$ eine Optimallösung des RMP und (π, μ) die zugehörige Duallösung ist, gilt:

$$\widetilde{c}\widetilde{\lambda}^* = (\pi, \mu)\widetilde{b}$$

Dem schwachen Dualitätssatz der linearen Optimierung zufolge, ist $\widetilde{c}\widetilde{\lambda}^*$ in diesem Fall also eine untere Schranke für den Zielfunktionswert des LMP und wegen der primalen Zulässigkeit auch eine obere Schranke. Damit ist die Lösung des RMP auch optimal für LMP.

Angenommen es gibt ein k, für das die Lösung des Problems (3.8) einen negativen Zielfunktionswert hat. Sei $\widetilde{x}^k$ also eine Lösung für (3.8) mit $c^k\widetilde{x}^k - \pi A^k\widetilde{x}^k - \mu_k < 0$. Dann hat die dieser Lösung zugehörige Spalte $\begin{pmatrix} A^k\widetilde{x}^k \\ e_k \end{pmatrix}$, mit Kosten $c^k\widetilde{x}^k$, negative reduzierte Kosten. Fügt man diese Spalte zur Nebenbedingungsmatrix und zum Kostenvektor des RMP hinzu, erhält man ein neues Restricted Master Problem, das dann reoptimiert wird. In diesem "abgeänderten" Pricing-Schritt wird also entweder die Optimalität der Lösung des RMP für das LMP nachgewiesen oder es wird eine neue Spalte für das LMP generiert, die negative reduzierte Kosten besitzt.

3.3 Untere Schranken

Wie schon erwähnt, ist der Zielfunktionswert des eingeschränkten Masterproblems eine obere Schranke für den Zielfunktionswert der LP-Relaxierung des Masterproblems. Bei Anwendung der Spaltenerzeugungsmethode erhält man zusätzlich nach jedem Pricing-Schritt

auch eine untere Schranke.

Sei c^{k^*} der optimale Zielfunktionswert des k-ten Pricing-Problems (3.8). z^*_{RMP} sei der optimale Zielfunktionswert des aktuellen eingeschränkten Masterproblems und z^*_{LMP} sei der (unbekannte) optimale Zielfunktionswert des LMP. Da der Wert des RMP nicht um mehr als $\sum_{k=1}^{K} c^{k^*}$ verbessert werden kann, gilt für z^*_{LMP}

$$z^*_{RMP} + \sum_{k=1}^{K} c^{k^*} \leq z^*_{LMP} \leq z^*_{RMP}.$$

Die obere Schranke z^*_{RMP} ist keine obere Schranke für den Zielfunktionswert des Masterproblems (6.20)-(6.22) sondern nur für die LP-Relaxierung des Problems. Die bis dahin beste gefundene untere Schranke hingegen ist auch eine untere Schranke für den Zielfunktionswert des Masterproblems. Durch diese untere Schranke kann also jederzeit die Güte der bis dahin besten gefundenen (ganzzahligen) Lösung bestimmt werden.

3.4 Branching

Die Lösung der LP-Relaxierung des Masterproblems ist nicht notwendigerweise ganzzahlig. Die Verzweigung des Problems soll den Lösungsraum so aufteilen, dass die aktuelle fraktionale Lösung ausgeschlossen wird. Wird das Branch-and-Bound-Verfahren für das Restricted Masterproblem mit den für die Lösung der LP-Relaxierung generierten Spalten gestartet, ist nicht garantiert, dass eine optimale (oder auch nur zulässige) Lösung für das Masterproblem gefunden wird. Es ist möglich, dass für die durch die Verzweigung des Problems neu entstandenen Masterprobleme Spalten mit negativen reduzierten Kosten existieren, die bisher noch nicht bekannt waren. Um eine optimale Lösung für das Masterproblem zu finden, muss in jedem Knoten des Branch-and-Bound-Baumes die LP-Relaxierung des Problems mittels Spaltenerzeugung gelöst werden. Daher wird das Verfahren auch Branch-and-Price genannt.

Standardbranchingregeln verzweigen auf den λ-Variablen des Masterproblems MP. Ist z.B. in der Lösung der LP-Relaxierung des Masterproblems eine Variable λ^k_t fraktional, würde der zulässige Bereich des Problems so aufgeteilt, dass in einem der beiden entstehenden Teilprobleme nur Lösungen mit $\lambda^k_t = 0$ und im anderen nur Lösungen mit$\lambda^k_t = 1$ zulässig sind.
Die zu λ^k_t gehörende Spalte $\begin{pmatrix} A^k x^{k,t} \\ e_k \end{pmatrix}$ des MP repräsentiert eine Lösung x^k_t des Subproblems (3.8). Mit $\lambda^k_t = 0$ wird diese Lösung ausgeschlossen. Wird die Struktur des k-ten Subproblems nicht verändert, ist es sehr gut möglich, dass beim nächsten Aufruf des Pricings genau diese Spalte wieder generiert wird. Um das auszuschließen, müsste im Subproblem für einen Knoten in Tiefe l des Branch-and-Bound-Baumes eine l-te beste Lösung gefunden werden.
Um diese Schwierigkeiten zu umgehen, ist es möglich, statt auf den λ-Variablen des MP

auf den Komponenten der Variablenvektoren x^k des Originalproblems IP zu verzweigen.

Eine Lösung der LP-Relaxierung des Masterproblems ist genau dann unzulässig für das Origianlproblem IP, wenn einer der Variablenvektoren

$$x^k = \sum_{t=1}^{T_k} x^{k,t} \lambda_t^k \tag{3.9}$$

eine fraktionale Komponente hat. Angenommen die Komponente i von x^k hat bei gegebener Lösung der LP-Relaxierung des Masterproblems den fraktionalen Wert α. Will man eine ganzzahlige Lösung für das Originalproblem zu finden, bietet es sich daher an, auf den Originalvariablen zu verzweigen. Das heißt, dass man in einem Abzweig

$$x_i^k = \sum_{t=1}^{T_k} x_i^{k,t} \lambda_t^k \leq \lfloor \alpha \rfloor$$

und im anderen

$$x_i^k = \sum_{t=1}^{T_k} x_i^{k,t} \lambda_t^k \leq \lceil \alpha \rceil$$

fordern muss.

Die Variablen, die die Branchingentscheidung verletzten, werden aus dem entsprechenden Masterproblem gelöscht.
Nach Möglichkeit sollten die Branchingentscheidungen im Pricing berücksichtigt werden, um auf diese Weise zu verhindern, dass Branchingentscheidungen verletztende Spalten erzeugt werden.

3.5 Farkas-Pricing

Die Löschung derjenigen Variablen aus dem Masterproblem, die eine Branchingentscheidung verletzten, kann dazu führen, dass das neue Masterproblem mit den verbliebenen Spalten unzulässig ist. In diesem Fall, braucht man ein Verfahren, mit dem sich zulässige Spalten für das Masterproblem erzeugen lassen, so dass das Masterproblem wieder zulässig wird. Das Farkas-Lemma liefert die Idee für ein solches Verfahren.
Ist die LP-Relaxierung des eingeschränkten Masterproblems

$$\begin{aligned} \min \tilde{c}^T \tilde{\lambda} \quad & \text{s.t.} \\ \widetilde{A}\widetilde{\lambda} &= \widetilde{b} \\ \widetilde{\lambda} &\geq 0 \end{aligned} \tag{3.10}$$

unzulässig, gibt es nach dem Farkas-Lemma einen Vektor π, so dass

$$\pi \widetilde{A} \leq 0 \quad \text{und} \quad \pi \widetilde{b} > 0. \tag{3.11}$$

Das zu (3.10) duale Problem

$$\max y^T \widetilde{b} \quad \text{s.t.} \tag{3.12}$$

$$y^T \widetilde{A} \leq \tilde{c} \tag{3.13}$$

ist daher unbeschränkt. Das Ziel besteht darin, eine für das Masterproblem (3.10) zulässige Spalte a zu finden, so dass

$$\pi a > 0.$$

Die Erweitert der Matrix $\widetilde{A}$ um eine solche Spalte a, entspricht dem Hinzufügen einer Nebenbedingung zum dualen Problem, so dass der Extremalstrahl π „abgeschnitten“ wird. Wird das duale Problem durch Hinzufügen dieser Nebenbedingung beschränkt, so ist das eingeschränkte Masterproblem nach Hinzufügen der Spalte a zur Matrix $\widetilde{A}$ zulässig.
Bleibt das duale Problem nach Hinzufügen der Nebenbedingung unbeschränkt, so existiert ein anderer Extremalstrahl $\hat{\pi}$ der (3.11) erfüllt.

Es soll demnach für jedes k $(k = 1, \ldots, K)$ und alle $x \in X_k^*$ überprüft werden, ob für die zugehörige Spalte $\begin{pmatrix} A^k x \\ e_k \end{pmatrix}$

$$\pi \begin{pmatrix} A^k x \\ e_k \end{pmatrix} > 0$$

erfüllt. Statt diese Ungleichung für jede Spalte des Masterproblems zu überprüfen, wird für jedes k das Optimierungsproblem

$$\min -\pi \begin{pmatrix} A^k x \\ e_k \end{pmatrix} x \quad \text{s.t.} \tag{3.14}$$

$$x \in X_k^*$$

gelöst. Hat eines dieser Probleme eine Lösung $\hat{x}$, sodass der Zielfunktionswert von (3.14) echt kleiner null ist, wird die Spalte $\begin{pmatrix} A^k \hat{x} \\ e_k \end{pmatrix}$ zur Nebenbedingungsmatrix des eingeschränkten Masterproblems hinzugefügt und reoptimiert. Wird das eingeschränkte Masterproblem dadurch zulässig, kann es mittels Spaltenerzeugung, wie in Abschnitt 3.2 beschrieben, gelöst werden.
Bleibt das eingeschränkte Masterproblem weiterhin unzulässig, werden die Probleme (3.14) erneut gelöst.

4 Set-Partitioning-Masterprobleme

Viele Probleme der kombinatorischen Optimierung lassen sich als Set-Partitioning-Probleme (Definition 2.9) formulieren. Auch Probleme der Tourenplanung sind häufig Set-Partitioning Probleme. In vielen Anwendungen des Branch-and-Price-Verfahrens ist das Masterproblem daher ein Set-Partitioning-Problem. An die zur Auswahl stehenden Teilmengen können zusätzliche Anforderungen gestellt werden. Beispielsweise kann gefordert werden, dass die Kapazität eines Fahrzeugs für Streugut in einer Tour nicht überschritten werden darf. Werden explizit Nebenbedingungen zu einem Set-Partitioning-Problem hinzugefügt, erhält man ein Set-Partitioning-Problem mit Nebenbedingungen.

Gegeben sei eine Grundmenge $E = \{1, \ldots, m\}$. Für die Variablen x_i^k gilt:

$$x_i^k = \begin{cases} 1 & \text{falls Teilmenge } k \text{ Element } i \in E \text{ enthält} \\ 0 & \text{sonst} \end{cases}$$

Der Vektor x^k sei der Inzidenzvektor der Teilmenge k. Die mit Teilmenge k assoziierten Kosten seien c_k. Ein allgemeines Set-Partitioning-Problem mit Nebenbedingungen hat die Form:

$$\begin{aligned} \min \sum_{k=1}^{K} c^k x^k \quad & \text{s.t} \\ \sum_{k=1}^{K} x^k &= \mathbb{1} \\ x^k &\in P_k \qquad k = 1, \ldots, K \\ x^k &\in \{0,1\}^{n_k} \qquad k = 1, \ldots, K \end{aligned} \tag{4.1}$$

Geht man davon aus, dass die Mengen P_k beschränkt sind, dann sind die Mengen $X_k^* = \{x \in X_k : x \text{ binär}\}$ endlich, also $P_k^* = \left\{x^{k,1}, \ldots, x^{k,T_k}\right\}$. Die Reformulierung des Problems ergibt sich wie in Abschnitt 3.1

$$\begin{aligned} \min \sum_{k=1}^{K} \sum_{t=1}^{T_k} c^k x^{k,t} \lambda_t^k \quad & \text{s.t.} \\ \sum_{k=1}^{K} \sum_{t=1}^{T_k} x^{k,t} \lambda_t^k &= \mathbb{1} \\ \sum_{t=1}^{T_k} \lambda_t^k &= 1 \\ \lambda_t^k &\in \{0,1\} \qquad k = 1, \ldots, K \; t = 1, \ldots, T_k \end{aligned} \tag{4.2}$$

4.1 Identische Bedingungen für die Teilmengen

Es kann nun vorkommen, dass die Beschränkungen für jede Teilmenge gleich sind. Will man zum Beispiel Touren für identische Fahrzeuge finden, so sind auch die Anforderungen an jede Tour gleich. Es ergibt sich dann folgendes Originalproblem

$$\begin{aligned} \min \sum_{k=1}^{K} c^k x^k \quad & s.t. \\ \sum_{k=1}^{K} x^k &= \mathbb{1} \\ x^k &\in P_k \qquad k = 1, \ldots, K \\ x^k &\in \{0,1\}^{n_k} \qquad k = 1, \ldots, K \end{aligned} \tag{4.3}$$

wobei

$$P_1 = \ldots = P_K$$

und

$$c^1 = \ldots = c^K$$

Wird Spaltenerzeugung zur Lösung der LP-Relaxierung des Masterproblems genutzt, so können die k-Subprobleme (3.8) zu einem Subproblem

$$\begin{aligned} \min cx - \pi x - \mu \quad & \text{s.t.} \\ x &\in P^* \end{aligned} \tag{4.4}$$

aggregiert werden. P^* bezeichne die Menge der zulässigen Teilmengen: $P^* = \{x \in X_1 : x \in \{0,1\}^{n_1}\} = \{x_1, \ldots, x^T\}$. Die aggregierte Form des Problems (4.2) ist:

$$\begin{aligned} \min \sum_{t=1}^{T} c^t x_t \lambda_t \quad & s.t. \\ \sum_{t=1}^{T} x_t \lambda_t &= \mathbb{1} \\ \sum_{t=1}^{T} \lambda_t &= K \\ \lambda_t &\in \{0,1\} \qquad t = 1, \ldots, T \end{aligned} \tag{4.5}$$

Die LP-Relaxierung kann für das Problem (4.5) mittels Spaltenerzeugung, wie in Abschnitt 3.2 beschrieben, gelöst werden.

4.2 Branching

Hat ein Problem identische Bedingungen für die Teilmengen und wird das Masterproblem (4.2) verwendet, kann, wie in Abschnitt 3.4 vorgestellt, auf den Variablen des Originalproblems (4.1) verzweigt werden. Für eine Lösung $\widetilde{\lambda}$ des Masterproblems (4.2) lassen sich die zugehörigen Werte der Originalvariablen berechnen.

$$\widetilde{x}_i^k = \sum_{t=1}^{T_k} x^{k,t} \lambda_t^k$$

Ist der Wert einer solchen Variablen fraktional, kann auf ihr verzweigt werden. Dabei ist in einem Abzweig gefordert, dass Element i in Teilmenge k enthalten ist und im anderen Abzweig, dass Element i nicht in Teilmenge k enthalten ist.
Bei identischen Restriktionen für Teilmengen ergeben sich aber exponentiell viele Lösungen, die sich nur dadurch unterscheiden, dass die Inzidenzvektoren der Teilmengen vertauscht wurden. Wird also auf einer fraktionalen Originalvariable $\widetilde{x}_i^k$ verzweigt und gibt es eine andere Teilmenge k' für die $\widetilde{x}_i^{k'}$ nicht fraktional ist, so wird sehr wahrscheinlich in einem anderen Knoten des Branch-and-Bound-Baumes eine Lösung auftauchen, für die die Teilmengen k und k' vertauscht wurden, d.h. in der neuen Lösung gilt $x^k = \widetilde{x}^{k'}$ und $x^{k'} = \widetilde{x}^k$. Das Branch-and-Bound-Verfahren ist in diesem Fall sehr ineffektiv, da sehr oft verzweigt werden muss, um eine fraktionale Lösung auszuschließen [2].

Wie im Abschnitt 4.1 beschrieben, kann man die Symmetrie des Problems durch die Aggregation der Teilprobleme berücksichtigen. Werden aggregierte Subprobleme verwendet, kann nicht auf den Originalvariablen verzweigt werde. Eine für Set-Partitioning-Probleme sehr weit verbreitete Branchingstrategie, die im Falle aggregierter Subprobleme anwendbar ist, stellt das Branchingverfahren von Ryan und Foster [27] dar.

4.2.1 Branchingregel von Ryan und Foster

Die Branchingstrategie von Ryan und Foster [27] ist eine Strategie für Set-Partitioning-Probleme, die auf folgendem Satz basiert.

Satz 4.1
Sei X die $0-1$-Nebenbedingungsmatrix eines Set-Partitioning Masterproblems und λ der Variablenvektor. Wenn eine Lösung $X\lambda = 1$ fraktional ist, also mindestens eine Komponente von λ fraktional ist, dann gibt es zwei Zeilen r und s in X, so dass

$$0 < \sum_{k: x_{rk}=1, x_{sk}=1} \lambda_p < 1 \tag{4.6}$$

Beweis (vgl. Barnhart et al. [2])
Sei $\lambda_{k'}$ eine fraktionale Variable. Die Zeile r sei eine beliebige Zeile mit $x_{rk'} = 1$. Weil λ zulässig ist, gilt: $\sum_{k=1}^{K} x_{rk}\lambda_k = 1$. Da $\lambda_{k'}$ fraktional ist, muss es eine andere Variable $\lambda_{k''}$

geben mit $0 < \lambda_{k''} < 1$ und $x_{rk''} = 1$. Da keine identischen Spalten in der Basis einer LP-Lösung vorkommen, muss es eine Zeile s geben, sodass entweder $x_{sk'} = 1$ oder $x_{sk''} = 1$ ist aber nicht beide. Die Nebenbedingungsmatrix X enthält also eine Submatrix der Form:

	k'	k''
r	1	1
s	0	1

Abbildung 4.1: Submatrix bei Ryan-Foster-Branchingkandidaten

Daher gilt:

$$\begin{aligned} 1 &= \sum_{k=1}^{K} x_{rk}\lambda_k \\ &= \sum_{k:x_{rk}=1} \lambda_k \\ &> \sum_{k:x_{rk}=1,x_{sk}=1} \lambda_p \end{aligned}$$

Die letzte Ungleichung gilt, da $\lambda_{k'}$ und $\lambda_{k''}$ nicht beide echt größer null sind. □

Daraus kann man dann die Branchingbedingungen

$$\sum_{k:x_{rk}=1,x_{sk}=1} \lambda_k = 1 \tag{4.7a}$$

$$\sum_{k:x_{rk}=1,x_{sk}=1} \lambda_k = 0 \tag{4.7b}$$

ableiten. Im ersten Abzweig fordert man, dass die Elemete r und s in der gleichen Teilmenge enthalten sind, im zweiten Abzweig fordert man, dass die beiden Elemente in unterschiedlichen Teilmengen enthalten sind. Wenn kein solches Paar von Zeilen gefunden werden kann, impliziert Satz 4.1, dass die Lösung des Masterproblems ganzzahlig ist. Da es nur endlich viele Paare von Zeilen gibt, muss der Branch-and-Bound-Algorithmus nach endlich vielen Branchingschritten terminieren. Jede Branchingentscheidung eliminiert bei dieser Branchingregel unter Umständen sehr viele Variablen des Masterproblems.

Da man im ersten Abzweig fordert, dass r und s von der selben Spalte überdeckt werden (d.h. in der selben Teilmenge enthalten sein sollen), muss für alle zulässigen Spalten in diesem Abzweig gelten:

$$x_{rk} = x_{sk} = 1 \qquad \text{oder} \qquad x_{rk} = x_{sk} = 0 \tag{4.8}$$

Im zweiten Abzweig gilt für alle zulässigen Spalten:

$$x_{rk} = 1, x_{sk} = 0 \qquad \text{oder} \qquad x_{rk} = 0, x_{sk} = 1 \qquad \text{oder} \qquad x_{rk} = x_{sk} = 0 \tag{4.9}$$

Statt diese Bedingungen explizit zum Masterproblem hinzuzufügen, können diejenigen Spalten, die zu unzulässigen Teilmengen gehören, aus dem Masterproblem gelöscht werden. Die Erzeugung der ausgeschlossenen Teilmengen muss dann im Pricing-Problem verhindert werden. Das hat den Vorteil, dass keine neuen Dualvariablen eingeführt werden müssen, die dann im Pricing-Problem berücksichtigt werden müssten. In den meisten Anwendungen ist es möglich, die Branchingentscheidungen im Pricing zu berücksichtigen, indem die Bedingungen (4.8) bzw. (4.9) als Nebenbedingungen des Subproblems eingeführt werden.

5 Das Resource-Constraint-Shortest-Path-Problem

In vielen Anwendungen des Branch-and-Price-Verfahrens, speziell bei Tourenplanungsproblemen, ist das Masterproblem häufig ein Set-Partitioning-Problem mit Nebenbedingungen, in dem die Variablen zu Touren gehören. Beispiele sind Tourenplanungsprobleme, bei denen mit einer Fahrzeugflotte eine Menge von Kunden mit Waren beliefert werden soll, Probleme der Müllabfuhr, in denen die Mülltonnen eines Straßenabschnitts geleert werden müssen und auch das Straßenwinterdienstproblem, in dem jeder Straßenabschnitt von einem Fahrzeug geräumt werden soll. Diese Touren werden in einem oder mehreren Subproblemen generiert, wobei die Subprobleme meistens Varianten des *Resource-Constraint-Shortest-Path-Problems (RCSPP)* sind. Im RCSPP will man von allen Wegen, die an einem Startknoten beginnen, an einem Zielknoten enden und die Bedingungen für die Ressourcen einhalten, den kürzesten durch ein Netzwerk finden. Eine Ressource ist dabei eine Größe, wie z.B. die Länge oder die Zeit, die sich entlang eines Weges gemäß einer gegebenen Funktion, der *Resource-Extension-Funktion (REF)*, verändert. Eine REF ist für jede Kante des Netzwerkes und jede Ressource definiert. Für gegebene Werte der Ressourcen am Anfangsknoten einer Kante berechnet die REF die Werte der Ressourcen im Endknoten dieser Kante. Die Bedingungen für die Ressourcen sind durch Intervalle gegeben, die die Werte der Ressourcen in jedem Knoten eines Weges beschränken. Solche Intervalle sind für alle Knoten des Netzwerkes definiert.

Das RCSPP wird normalerweise durch Dynamische Programmierung, genauer durch einen Labeling-Algorithmus gelöst. Eine ausführliche Beschreibung des RCSPP und seiner Varianten und verschiedenen Lösungsverfahren sowie Details zur Implementierung eines Labeling-Algorithmus für das RCSPP findet man in Irnich und Desaulniers [13]. Dieses Kapitel soll eine kurze Einführung in das RCSPP geben. Im Folgenden werden auch Teiltouren, also Pfade, die nicht den selben Anfangs- und Endknoten haben, als Tour bezeichnet.

5.1 Resource-Constraints und die Resource-Extension-Funktion

Das RCSPP wird auf einem Graph $G(V, A, g)$ formuliert. Im RCSPP wird eine „zulässige“ Tour durch den Graphen gesucht. Man kann dabei zwischen der Zulässigkeit einer Tour bezüglich der Ressourcen und bezüglich ihrer Struktur unterscheiden. Bedingungen für die Struktur einer Tour kann man z.B. durch Branchingentscheidungen erhalten, die bestimmte Touren ausschließen. Wie solche Bedingungen im RCSPP berücksichtigt werden können, wird im nächsten Abschnitt besprochen. Dieser Abschnitt beschäftigt sich mit der Zulässigkeit von Touren bezüglich der Ressourcen.

Nebenbedingungen für Ressourcen können durch ihren „Verbrauch“ auf einer Kante a und zulässige Intervalle beschrieben werden, z.B. die Zeit t_a und Zeitintervalle $[t_i^1, t_i^2]$. Sei R die Anzahl der Ressourcen. Der Vektor $T = (T^1, \ldots, T^R)^T$ heißt Ressourcenvektor und die Komponenten des Vektors heißen Ressourcenvariablen.

Für zwei Ressourcenvektoren u und v ist das Intervall $[u, v]$ definiert als die Menge

$$\left\{T \in \mathbb{R}^R : u \leq T \leq v\right\}.$$

An einem Knoten i beschreibt $[u_i, v_i]$ das Intervall für die zulässigen Ressourcenvektoren. Der Verbrauch der Ressourcen entlang einer Kante $a \in A$ ist durch einen Vektor $f_a = (f_a^r)_{r=1}^R$ von Funktionen gegeben, den Resource-Extension-Funktionen (REF). Eine REF $f_a^r : \mathbb{R}^R \to R$ beschreibt den Verbrauch der Ressource r entlang Kante a und hängt von einem Ressourcenvektor T_i ab, der dem Verbrach der Ressourcen vom Startknoten bis zum Knoten i entspricht, wobei i der Anfangknoten der Kante a ist. In einfachen REF hängt der Verbrauch einer Ressource r nur von der Komponente r des Ressourcenvektors T_i ab (es gibt keine Abhängigkeiten zwischen den Ressourcen). In diesem Fall hat eine REF die Form

$$f_a^r(T_i) = T_i^r + t_a^r \tag{5.1}$$

wobei t_a^r eine Konstante ist.

Gegeben sei eine Tour $p = (v_1, a_1, v_2, \ldots, a_{n-1}, v_n)$. In einer Tour können sowohl Knoten als auch Kanten des Graphen mehrfach vorkommen, daher wird mit v_i der Knoten an Position i und mit a_i die Kante an Position i der Tour notiert. Die Tour p ist zulässig bezüglich der Ressourcen, falls es Ressourcenvektoren $T_i \in [u_i, v_i]$ für alle Positionen $i = 1, \ldots, n$ gibt, sodass $f_{a_i}(T_i) = T_{i+1}$ für $i = 1, \ldots, n-1$ gilt. Die Menge aller bezüglich der Ressourcen zulässigen Pfade von einem Knoten u zu einem Knoten v des Graphen, sei $\mathcal{F}(u, v)$.

5.2 Berücksichtigung von Bedingungen an die Struktur einer Tour

Bedingungen an die Pfadstruktur können Anforderungen an die Zulässigkeit einer Tour modellieren, die nicht durch die Ressourcen gegeben sind. Zusätzliche Anforderungen an die Tour können in einem Branch-and-Price-Verfahren durch die Berücksichtigung von Branchingentscheidungen in den Subproblemen entstehen. Wird eine Tour aus dem Restricted-Master-Problem gelöscht, so darf sie im Pricing-Problem nicht wieder generiert werden. Für das Straßenwinterdienstproblem ist außerdem auch der Fall interessant, der im Subproblem nur einfache Touren zuläßt, also Touren, in denen jede Kante des Graphen höchstens einmal vorkommen darf.

Im Folgenden wird erläutert, wie man Ryan-Foster-Branchingentscheidungen (siehe S.24) im RCSPP berücksichtigen kann.
Ist ein Graph gegeben, in dem Kanten Aufgaben repräsentieren, wie zum Beispiel die Räumung eines Straßenabschnittes von Schnee oder die Leerung von Mülltonnen in einem Straßenabschnitt, dann repräsentiert eine Tour durch den Graphen eine Abfolge von Aufgaben. Jede Aufgabe soll in genau einer Tour erledigt werden, das heißt jede Tour soll einfach sein. Die Spalten im Masterproblem ensprechen Inzidenzvektoren von Aufgaben. Da jede Aufgabe genau einmal erledigt werden soll, enthält das Masterproblem für jede

Aufgabe eine Set-Partitioning-Bedingung.Sollen alle Touren einfach sein, so ist die Menge der zulässigen Touren

$$\mathcal{G}_e = \{p : p \text{ einfach}\}.$$

In der Branchingregel von Ryan und Foster wird für zwei Aufgaben a_1 und a_2 in einem Abzweig gefordert, dass beide Aufgaben in der selben Tour erledigt werden, im anderen Abzweig, dass sie in zwei verschiendenen Touren erledigt werden. Die Menge der, bezüglich dieser Bedingungen, zulässigen Touren $\mathcal{G}$ ist im ersten Fall

$$\mathcal{G}_s = \{p : a_1 \in p \text{ und } a_2 \in p\} \cup \{p : a_1 \notin p \text{ und } a_2 \notin p\}$$

im zweiten Fall

$$\mathcal{G}_d = \{p : \text{entweder } a_1 \notin P \text{ oder } a_2 \notin p\}.$$

Sowohl die Bedingung, dass alle Touren einfach sein sollen, als auch die Ryan-Foster-Branchingentscheidungen können mit Hilfe zusätzlicher Ressourcen modelliert werden.
Um sicherzustellen, dass eine Tour einfach ist, wird für jede Kante $a \in A$ eine Ressource r_a und eine Ressourcenvariable T^{r_a} eingeführt. Die REF für diese Ressource entlang einer Kante $e \in A$ mit $g(e) = (i, j)$ lautet

$$f_e^{r_a}(T_i^a) = \begin{cases} T_i^{r_a} + 1 & \text{falls } e = a \\ T_i^{r_a} & \text{sonst} \end{cases} \tag{5.2}$$

wobei in keinem Knoten des Graphen nur Werte im Intervall $[0, 1]$ für jede dieser Ressourcenvariablen zulässig ist. Damit ist sichergestellt, dass keine Kante mehr als einmal vorkommen kann.

Für die Berücksichtigung der Branchingentscheidung, dass zwei Aufgaben(Kanten) a_1 und a_2 zusammen in einer Tour erledigt werden müssen, führt man eine neue Ressource it_{a_1,a_2} ein. Die REF für diese Ressource entlang einer Kante a mit Anfangsknoten i ist gegeben durch

$$f_a^r(T_i^{it_{a_1,a_2}}) = \begin{cases} T_i^{it_{a_1,a_2}} + 1 & \text{falls } a = a_1 \\ T_i^{it_{a_1,a_2}} - 1 & \text{falss } a = a_2 \\ T_i^{it_{a_1,a_2}} & \text{sonst,} \end{cases} \tag{5.3}$$

die zulässigen Intervalle in den Knoten i für diese Ressource sind gegeben durch

$$[0, 0] \quad \text{für } i = s, t \tag{5.4}$$
$$[-1, 1] \quad \text{für } i \in V \setminus \{s, t\} \tag{5.5}$$

wobei s der Start- und t der Zielknoten der Tour ist.

Sollen a_1 und a_2 nicht in der selben Tour enthalten sein, führt man eine Ressource $fbit_{a_1,a_2}$ ein. Die REF für diese Ressource ist

$$f_a^r(T_i^r) = \begin{cases} T_i^r + 1 & \text{falls } a = a_1 \\ T_i^r - 1 & \text{falss } a = a_2 \\ \infty & \text{falls } a \in \{a_1, a_2\} \text{ und } T_i^r \neq 0 \\ T_i^r & \text{sonst.} \end{cases} \tag{5.6}$$

Die zulässigen Intervalle in den Knoten i für diese Ressource sind gegeben durch

$$[0, 0] \quad \text{für } i = s \tag{5.7}$$
$$[-1, 1] \quad \text{für } i \in V \setminus \{s\}. \tag{5.8}$$

Im folgenden bezeiche $\mathcal{G}$ die Menge der Touren, die bezüglich aller Bedingungen an die Pfadstruktur zulässig sind.

5.3 Dynamische Programmierung und Labeling-Alorithmus

Lösungsverfahren der Dynamischen Programmierung für das RCSPP konstruieren systematisch neue Touren, in dem ausgehend von der Teiltour $p = (s)$, iterativ Touren, mithilfe der REF, entlang der Kanten in alle zulässigen Richtungen erweitert werden. An jedem Knoten wird überprüft, ob der Ressourcenvektor in dem an diesem Knoten zulässigen Intervall liegt. Damit nicht alle zulässigen Touren generiert werden müssen, werden Touren, die nicht zu einer optimalen Tour erweitert werden kömnen, mithilfe einer Dominanzfunktion verworfen.

Ein Label ist das Tripel aus aktuellem Knoten, aktuellem Ressourcenvektor und derjenigen Kante über die das Label zum aktuellen Knoten erweitert wurde. Die Label werden zur Kodierung von Teiltouren verwendet. Ein Label für einen Pfad $p = (v_1, a_1, v_2, \ldots, a_{p-1}, v_p)$ ist direkt mit dem Label der vorangegangenen Teiltour $q = (v_1, a_1, v_2, \ldots, a_{p-2}, v_{p-1})$ verknüpft. Ein Pfad $p = (v_1, a_1, v_2, \ldots, a_{n-1}, v_n)$ ist eine *zulässige Erweiterung* für einen Pfad $q = (w_1, e_1, w_2, \ldots, e_{m-1}, w_m)$, falls $(q, p) = (w_1, e_1, \ldots, e_{m-1}, w_m, a_1, \ldots, a_{n-1}, v_n) \in \mathcal{F}(w_1, v_n) \cap \mathcal{G}$. Die Menge aller zulässigen Erweiterungen eines Pfades q sei

$$\mathcal{E}(q) = \{p : (q, p) \in \mathcal{F}(w_1, v(p)) \cap \mathcal{G}\},$$

wobei $v(p)$ der Endknoten des Pfades p ist.

Gegeben seien zwei verschiedene Touren p und q, die den gleichen Endknoten haben, das heißt $v(p) = v(q)$, dann dominiert die Tour p die Tour q genau dann, wenn

$$\mathcal{E}(p) \supseteq \mathcal{E}(q) \tag{5.9}$$

gilt. Das heißt, Tour p dominiert Tour q, falls jede zulässige Erweiterung der Tour q auch eine zulässige Erweiterung der Tour p ist. Eine Tour, die von keiner anderen Tour dominiert

wird, heißt *pareto-optimal*.
Dominiert also Tour p Tour q so kann Tour q gelöscht werden, denn sie kann nicht zu einer pareto-optimalen Tour im Zielknoten erweitert werden.

Die Ressourcenvektoren der Touren p und q im Endknoten der Touren $v(p) = v(q)$ sei gegeben durch $T(p)$ bzw. $T(q)$. Sind keine Anforderungen an die Struktur der Tour gestellt und haben alle REF die Form (5.1), dann gilt (5.9), falls $T(p) < T(q)$ ist. In diesem Fall ist jede bezüglich der Ressourcen zulässige Erweiterung für q auch zulässig für p.

Sind nur einfache Touren zulässig, folgt aus der Bedingung $T(p) < T(q)$ nicht notwendigerweise $\mathcal{E}(p) \supset \mathcal{E}(q)$, denn Touren $p \in \mathcal{G}$ können nur über Kanten erweitert werden, die noch nicht in p enthalten sind.

Sei $A(P)$ die Menge der Kanten des Graphen $G(V, A, g)$, die in der Tour P enthalten sind. Damit eine Tour P eine Tour Q dominieren kann, muss $A(p) \subseteq A(q)$ und $T(p) < T(q)$ gelten. Die Menge $A(p)$ kann durch die oben beschriebenen Ressourcen r_a für jede Kante $a \in A$ modelliert werden. Damit $A(p) \subseteq A(q)$ gilt, muss für die neu eingeführten Ressourcenvariablen r_a gelten:

$$T^{r_a}(p) \leq T^{r_a}(q) \quad \text{für alle } a \in A.$$

Sind nur Touren zulässig in denen zwei Kanten a_1 und a_2 gemeinsam vorkommen, z.B. in einem Branch-and-Price-Algorithmus mit Ryan-Foster-Brachingregel, können Touren $P \in \mathcal{G}$ nur so erweitert werden, dass sie im Endknoten der Tour entweder sowohl a_1 als auch a_2 enhalten oder keine von beiden Kanten. Für p und q gelte $T(p) < T(q)$. Falls p und q die Kante a_1 enthalten aber nicht a_2, so sind für beide Touren nur Erweiterungen zulässig, die a_2 enthalten nicht aber a_1, es gilt also $\mathcal{E}(p) \subseteq \mathcal{E}(q)$. Enthalten beide die Kante a_2 aber nicht a_1, gilt das selbe Argument. Enthalten beide a_1 und a_2, sind für beide Touren nur Erweiterungen zulässig, die weder a_1 noch a_2 enthalten. Sind weder a_1 noch a_2 in p und q enthalten, sind für beide Touren Erweiterungen zulässig, die entweder beide Kanten oder keine von beiden enthalten. Man kann für jede solche Bedingung eine Ressource $it(a_1, a_2)$ (s. oben) einführen. Falls $T^{it(a_1,a_2)}(p) = T^{it(a_1,a_2)}(q)$, sind die Kanten a_1 und a_2 beide gleich oft in den Touren enthalten.

Sind nur Touren zulässig, in denen zwei Kanten a_1 und a_2 nicht gemeinsam vorkommen, muss $T(p) < T(q)$ gelten und die Tour p darf entweder keine der beiden Kanten a_1 und a_2 enthalten oder Touren p und q müssen die selbe Kante a_1 oder a_2 enthalten, damit $\mathcal{E}(p) \supseteq \mathcal{E}(q)$ gilt.

Seien also p und q Touren mit $T(p) < T(q)$. Enthält p keine der beiden Kanten, so sind für p Erweiterungen zulässig, die höchstens eine der beiden Kanten enthalten. Enthält q auch keine der Kanten, sind für p und q die selben Erweiterungen zulässig, d.h. $\mathcal{E}(p) \supseteq \mathcal{E}(q)$. Enthält q schon eine der Kanten, sind für q nur Erweiterungen zulässig, die keine der beiden Kanten a_1 oder a_2 enhalten, d.h $\mathcal{E}(p) \supset \mathcal{E}(p)$. Enthalten p und q die selbe Kante z.B.

a_1, so sind für beide Touren nur Erweiterungen zulässig, die keine der Kanten enthalten, also $\mathcal{E}(p) \supseteq \mathcal{E}(q)$. Wird für eine solche Branchingentscheidung eine Ressource $fbit(a_1, a_2)$ eigeführt (s. oben), so kann Tour p Tour q nur dominieren, falls $T^{fbit(a_1,a_2)}(p) = 0$ oder $T^{fbit(a_1,a_2)}(p) = T^{fbit(a_1,a_2)}(q)$.

Die Idee des Algorithmus besteht darin, in jeder Iteration ein Label L aus den zu bearbeitenden Labeln auszusuchen. Der Knoten i sei der aktuelle Knoten, an dem das Label L gespeichert ist. Das Label L wird mit allen anderen Labeln, die am selben Knoten gespeichert sind, auf Dominanz überprüft. Wird L von einem anderen Label dominiert, wird L gelöscht. Falls L von keinem anderen Label dominiert wird, wird L entlang aller Kanten $a \in \delta^+(i)$ erweitert, zu der Liste der bearbeiteten Label hinzugefügt und anschließend aus der Liste der noch zu bearbeitenden Label gelöscht. Jedes der neu enstandenen Label wird auf Zulässigkeit überprüft. Ist das neue Label zulässig, wird es zur Liste der zu bearbeitenden Label hinzugefügt. Ist das neue Label unzulässig, wird es verworfen. Gibt es keine zu bearbeitenden Label mehr, stoppt der Algorithmus und konstruiert aus der Liste der bearbeiteten Label alle pareto-optimalen Touren mit vorgegebenem Startknoten s und Zielknoten t.

Enthält der Graph $G(V, A, g)$ Kreise und soll ein Label-Setting-Algorithmus verwendet werden, benötigt man zumindest eine Ressource mit strikt positivem Verbrach auf jeder Kante des Graphen. D.h. es gilt $f_a^r(T_i) > T_i^r$ für alle Kanten $a \in A$ und alle zulässigen Ressourcenvektoren T_i. Eine solche Ressource wäre zum Beispiel die Kantenlänge. In diesem Fall kann im Labeling-Algorithmus jeweils die Tour q für die Erweiterung ausgesucht werden,die den kleinsten Wert $T^r(q)$ besitzt. Damit ist garantiert, dass $T^r(q, p) > T^r(q)$ für jede Erweiterung (q, p) von q gilt. Ein schon einmal erweitertes Label kann somit nie von einem anderen Label dominiert werden. Details zum Labeling-Algorithmus für das RCSPP sind in Irnich und Desaulniers [13] nachzulesen.

6 Das Straßenwinterdienst Problem

Das Amt für Straßen- und Verkehrswesen (ASV) Dillenburg ist für den Bau, den Erhalt und den Unterhalt der „klassifizierten“ Straßen im Lahn-Dill-Kreis und dem Kreis Limburg-Weilburg zuständig. Zu den klassifizierten Straßen gehören Autobahnen, Bundes-, Landes-, und Kreisstraßen. In Abb. 6.1 sind die Streckenlängen der klassifizierten Straßen aufgelistet.

Straßentyp	Länge
Autobahnen	158 km
Bundesstraßen	260 km
Landesstraßen	728 km
Kreisstraßen Lahn-Dill	264 km
Kreisstraßen Limburg-Weilburg	245 km

Abbildung 6.1: Streckenlängen Straßennetz

Weitere Informationen zu dem hier behandelten Straßennetz und den verwendeten Daten erhält man bei [12]. Dort kann man auch eine Netzknotenkarte des Straßennetzes herunterladen.

Da für das Räumen und Streuen der Autobahnen eine eigene Straßenmeisterei (SM) zuständig ist, bleiben diese unbetrachtet. Auch die Kreisstraßen im Kreis Limburg-Weilburg bleiben ausgeklammert, da sie vom Kreis selbst geräumt und gestreut. Betrachtet werden somit alle Bundes- und Landesstraßen in den Kreisen Lahn-Dill und Limburg-Weilburg sowie die Kreisstraßen im Kreis Lahn-Dill. Für deren Räumung und Streuung sind die Straßenmeistereien (SM) Dillenburg, Brechen, Solms und Weilburg zuständig. Die Autobahnen und die Straßen im Kreis Limburg-Weilburg dürfen durchfahren werden, müssen aber nicht geräumt werden. Momentan ist das Straßennetz in vier Distrikte aufgeteilt. Jede Straßenmeisterei ist für den Straßenwinterdienst in einem der Distrikte zuständig. Die derzeitige Aufteilung der Distrikte wird in dieser Arbeit vernachlässigt, denn es ist sehr wahrscheinlich, dass im Laufe der Zeit aus Kostengründen Straßenmeistereien zusammengelegt werden.

Die Touren für die Straßenwinterdienstfahrzeuge sollen von den Straßenmeistereien so geplant werden, dass alle Straßen innerhalb von drei Stunden geräumt werden können.

Zur Vereinfachung des Problems wird davon ausgegangen, dass die Anzahl der zur Verfügung stehenden Fahrzeuge nicht beschränkt ist, und dass alle Fahrzeuge die gleichen Eigenschaften haben. Die Anzahl der zur Räumung des Straßennetzes benötigten Fahrzeuge ergibt sich in der Optimallösung aus der Anzahl der Touren. Weiter wird angenommen, dass die Straßen nur geräumt und nicht gestreut werden müssen, so dass die Streusalzka-

pazitäten der Fahrzeuge vernachlässigt werden können.

Es wird eine Durchschnittsgeschwindigkeit von 30 km/h der Fahrzeuge während eines Räumeinsatzes unterstellt. Da alle Straßen innerhalb von drei Stunden geräumt sein sollen, kann ein Fahrzeug während eines Räumeinsatzes höchstens 90 km zurücklegen. Außerdem muss jede Spur einzeln geräumt werden und die Fahrzeuge dürfen die Spuren nur in Fahrtrichtung durchfahren.

Alle Richtungsspuren der Straßen im Zuständigkeitsbereich der Straßenmeistereien sollen von genau einem Fahrzeug geräumt werden. Natürlich ist aber auch erlaubt, dass ein Fahrzeug eine Straße nur durchfährt. Es kann also zwischen den Aufgaben „Räumen“ und „Durchfahren“ unterschieden werden.
Die Tour jedes Fahrzeugs muss an einem Depot beginnen und am selben Depot enden.
Jede Tour eines Fahrzeuges kann dann als Abfolge von Aufgaben verstanden werden.
Das Problem die Spuren aller Straßen innerhalb von drei Stunden mit genau einem Fahrzeug zu räumen, kann als ganzzahliges Programm auf Grundlage eines Graphen formuliert werden.

6.1 Modellierung als Graph

Das Problem kann auf Grundlage eines gerichteten Graphen $G(V, A, g)$ formuliert werden. Jede Kante dieses Graphen steht für eine Aufgabe, die ein Fahrzeug auf einem Straßenabschnitt (zwischen zwei Kreuzungen) erledigt. Die Knoten des Graphen entsprechen den Kreuzungen im Straßennetz. Zusätzlich werden für jede Straßenmeisterei zwei Knoten eingeführt, ein Startknoten und ein Zielknoten. Der Startknoten d_s^i des Depots $d^i,\ i = 1, 2, 3, 4$ hat Eingangsgrad null. Der Zielknoten d_t^i hat Ausgangsgrad null. Außerdem wird zwischen jedem Startdepotknoten d_s^i und jedem Zieldepotknoten d_t^i eine Kante eingefügt um leere Touren zu ermöglichen.

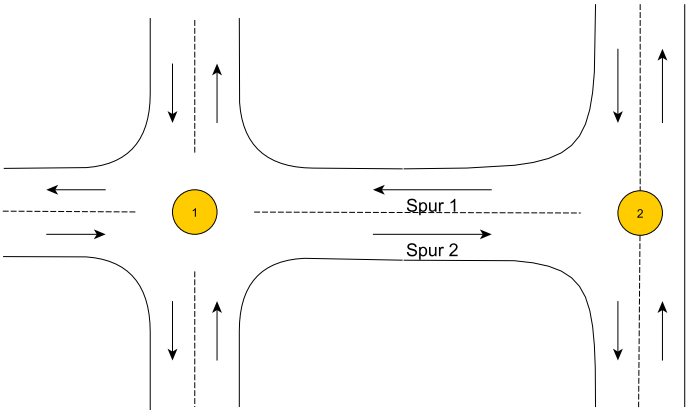


Abbildung 6.2: Kreuzung

Zwischen den in Abb. 6.2 dargestellten Kreuzungen gibt es in jede Richtung eine Spur. Sollen diese beiden Spuren geräumt werden, gibt es im Graph G zwei Kanten a_1 und a_2 mit $g(a_1) = (1, 2)$ und $g(a_2) = (2, 1)$, die die Aufgaben „Räumen“ der jeweiligen Spur repräsentieren. Außerdem gibt es in G zwei weitere Kanten d_1 und d_2 mit $g(d_1) = (1, 2)$ und $g(d_2) = (2, 1)$, die das „Durchfahren“ des jeweiligen Straßenabschnitts repräsentieren.

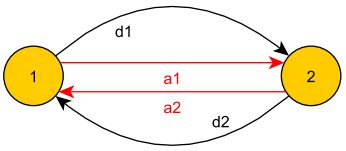


Abbildung 6.3: Modellierung als Graph

Jeder Straßenabschnitt hat eine Länge. Die Funktion $l : A \to R_+$ ordnet jeder Kante des Graphen die Länge des zugehörigen Straßenabschnitts zu. Für $a \in A$ mit $g(a) = (ij)$ gilt

$$l(a) = l_a = l_{ij}.$$

Die Kante zwischen den Start- und Zieldepotknoten hat die Länge Null.

Die Menge $A_R \subset A$ sei die Menge derjenigen Kanten, die die Räumung einer Spur repräsentieren. Mit $\bar{A}_R$ wird die Menge der Kanten, die das einfache Durchfahren eines Straßenabschnitts repräsentieren, bezeichnet. F sei die Menge aller Fahrzeuge. Es wird davon ausgegangen, dass die Anzahl der zur Verfügung stehenden Fahrzeuge beliebig groß ist.

Das Ziel besteht darin, Touren durch den Graphen zu finden, sodass jede Kante $a \in A_R$ in genau einer Tour vorkommt. Dabei soll keine Tour die Gesamtlänge von 90 km

überschreiten.
D sei die Anzahl der Depots (Straßenmeistereien). Der Anfangsknoten einer Tour muss einem der Startdepotknoten $d_s^i,\ i = 1, \ldots, D$ entsprechen. Der Endknoten muss der zum Startknoten d_s^i gehörende Zielknoten d_t^i sein.

6.2 Formulierung als ganzzahliges Programm

Das Straßenwinterdienstproblem kann als ganzzahliges Programm formuliert werden. Es wird für jede Kante $a \in A_R$ und jedes Fahrzeug $f \in F$ eine Variable $x_{af} \in \{0, 1\}$ eingeführt, mit

$$x_{af} = \begin{cases} 1 & \text{falls } a \text{ in der Tour für Fahrzeug } f \text{ enthalten ist} \\ 0 & \text{sonst.} \end{cases}$$

Für jede Kante $a \in \bar{A_R}$ und jedes Fahrzeug $f \in F$ wird eine Variable $x_{af} \in \mathbb{Z}_+$ definiert, die angibt, wie oft das Fahrzeug f den zu a gehörenden Straßenabschnitt durchfährt.

Für jeden Knoten v in einer Tour (s. Def. 2.12) muss der Eingangsgrad gleich dem Ausgangsgrad sein.
Es muss für jedes Fahrzeug $f \in F$ und für jeden Knoten $v \in V$

$$\sum_{a \in \delta^+(v)} x_{af} = \sum_{a \in \delta^-(v)} x_{af} \tag{6.1}$$

gelten. Die Tour jedes Fahrzeugs soll an einem Depot beginnen und am selben Depot wieder enden. Es muss daher für jedes Fahrzeug $f \in F$

$$\sum_{i=1}^{D} \sum_{a \in \delta^+(d_s^i)} x_{af} = 1 \tag{6.2}$$

und für i=1,. . . ,D

$$\sum_{a \in \delta^+(d_s^i)} x_{af} = \sum_{a \in \delta^-(d_t^i)} x_{af} \tag{6.3}$$

gelten. Diese Bedingungen sind aber noch nicht ausreichend, um den Zusammenhang jeder Tour zu sicherzustellen. Das macht Abb. 6.4 deutlich. Obwohl die Kantenfolge (a_1, a_2, a_3, a_4) sowohl die Bedingung (6.1) für alle Knoten erfüllt als auch die Bedingung (6.2) erfüllt ist, bilden die Kanten jedoch keine Tour.

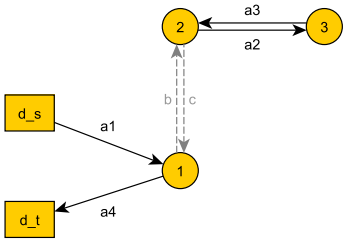


Abbildung 6.4

Es gibt verschiedene Möglichkeiten den Zusammenhang einer Tour zu gwährleisten. Eine Möglichkeit besteht darin, für jede Teilmenge $S \subset V \setminus \{d_s^i, d_t^i : i = 1, \dots D\}$ der Knoten (ohne die Depots) zu fordern, dass falls $e \in (S \times S) \cap A$ in der Tour enthalten ist, so muss auch mindestens eine Kante aus $\delta^+(S)$ enthalten sein.

$$x_{ef} - \sum_{a \in \delta_S^+} x_{af} \leq 0 \tag{6.4}$$

In dem Beispiel aus Abb. 6.4 bewirkt die Bedingung (6.4), dass die Tour auch die Kante c enthalten muss. Aus Bedingung (6.1) folgt dann, dass auch die Kante b in der Tour enthalten sein muss. Es ergibt sich also eine zusammenhängende Tour.
Durch die Formulierung (6.4) der sogenannten Subtour-Elimination-Constraints erhält man exponentiell viele Nebenbedingungen.

Die Summe der Längen aller Kanten der Tour soll die maximale Länge $l_{max} = 90$ km nicht überschreiten, d.h.

$$\sum_{a \in A} l_a x_{af} \leq l_{max}.$$

Das Ziel besteht darin, die insgesamt von allen Fahrzeugen zurückgelegte Strecke zu mini-

mieren. Das Straßenwinterdienstproblems läßt sich wie folgt als IP formulieren

$$\min \sum_{f \in F} \sum_{a \in A} l_a x_{af} \quad \text{s.t.} \tag{6.5}$$

$$\sum_{f \in F} x_{af} = 1 \qquad a \in A_R \tag{6.6}$$

$$\sum_{a \in A} l_a x_{af} \leq l_{max} \qquad f \in F \tag{6.7}$$

$$\sum_{a \in \delta^+(v)} x_{af} = \sum_{a \in \delta^-(v)} x_{af} \qquad f \in F, v \in V \tag{6.8}$$

$$\sum_{i=1}^{4} \sum_{a \in \delta^+(d_s^i)} x_{af} = 1 \qquad f \in F \tag{6.9}$$

$$\sum_{a \in \delta^+(d_s^i)} x_{af} = \sum_{a \in \delta^-(d_t^i)} x_{af} \qquad f \in F, i = 1, \ldots, D \tag{6.10}$$

$$x_{ef} - \sum_{a \in \delta^+(S)} x_{af} \leq 0 \qquad S \subset V \setminus \left\{ d_s^i, d_t^i : i \leq D \right\}, e \in (S \times S) \cap A, f \in F \tag{6.11}$$

$$x_{af} \in \{0, 1\} \qquad a \in A_R, f \in F \tag{6.12}$$

$$x_{af} \in \mathbb{Z}_+ \qquad a \in \bar{A_R}, f \in F \tag{6.13}$$

Die Nebenbedingungen (6.7)-(6.13) des Problems sind nach Fahrzeugen trennbar. Die Nebenbedingungsmatrix hat daher die in Abb. 3.1 (s. S. 14) dargestellte Struktur. Die Nebenbedingungen (6.6) verbinden die Nebenbedingungsblöcke der einzelnen Fahrzeuge. Die Bedingungen (6.7)-(6.13) definieren für jedes Fahrzeug den zulässigen Bereich eines Resource-Constrained-Shortest-Path-Problems, wobei die Kanten $a \in A_R$ wegen der Bedingungen (6.12) nur einmal in einer Tour enthalten sein dürfen.

6.3 Reformulierung

Folgende Notationen werden für die Reformulierung des Problems verwendet. Wenn p eine Tour durch den Graphen bezeichnet, dann sei der Vektor x_p der Inzidenzvektor dieser Tour, der für jede Kante $a \in A_R$ einen Eintrag

$$x_{pa} = \begin{cases} 1 & \text{falls } a \in A_R \text{ und } a \text{ in der Tour p enthalten ist} \\ 0 & \text{falls } a \in A_R \text{ und } a \text{ nicht in Tour } p \text{ enthalten ist} \\ k & \text{falls } a \in \bar{A_R} \text{ und } a \text{ } k\text{-mal in Tour } p \text{ enthalten ist} \\ 0 & \text{falls } a \in \bar{A_R} \text{ und } a \text{ nicht in Tour } p \text{ enthalten ist} \end{cases}.$$

enthält.

Sei Z^f die Menge aller für Fahrzeug f zulässigen Touren. Das heißt Z^f ist die Menge aller Touren, die an einem Depot beginnen und am selben Depot enden und die Gesamtlänge von l_{max} nicht überschreiten. P^f sei die Menge der Inzidenzvektoren für diese Touren.

Für jedes Fahrzeug f sind

$$Z^f = \left\{p_1^f, \ldots, p_n^f\right\}$$

und

$$P^f = \left\{x_{p_1}^f, \ldots, x_{p_1}^f\right\}$$

endliche Mengen.

Auf dieser Grundlage läßt sich das Straßenwinterdienstproblem auch folgendermaßen formulieren

$$\begin{aligned} \min \sum_{f\in F}\sum_{a\in A} l_a x_a^f \quad & \text{s.t.} \\ \sum_{f\in F} x_a^f = 1 \qquad & a \in A_R \\ x^f = (x_a^f)_{a\in A} \in P^f \qquad & f \in F \end{aligned} \tag{6.14}$$

Da P^f endlich ist, gibt es nach Satz 2.2 für jeden Vektor x^f eine Darstellung

$$\begin{aligned} x^f &= \sum_{p\in Z^f} x_p^f \lambda_p^f \\ \sum_{p\in Z^f} \lambda_p^f &= 1 \\ \lambda_p^f \in \{0,1\} \qquad & p \in Z^f, f \in F \end{aligned}$$

Durch Substitution in (6.14) ergibt sich folgendes Masterproblem:

$$\begin{aligned} \min \sum_{f\in F}\sum_{a\in A}\sum_{p\in Z^f} l_a x_{pa}^f \lambda_p^f \quad & \text{s.t.} \\ \sum_{f\in F}\sum_{p\in Z^f} x_{pa}^f \lambda_p^f = 1 \qquad & a \in A_R \\ \sum_{p\in Z^f} \lambda_p^f = 1 \qquad & f \in F \\ \lambda_p^f \in \{0,1\} \qquad & p \in Z^f, f \in F \end{aligned} \tag{6.15}$$

Das Straßenwinterdienstproblem besteht nun in der Auswahl einer Tour $p^f \in Z^f$ für jedes Fahrzeug, sodass jede Kante $a \in A_R$ in genau einer der augewählten Touren enthalten ist

und die Summe aller Tourlängen minimal ist.

Weil davon ausgegangen wird, dass alle Fahrzeuge identisch sind, entsprechen sich auch die Mengen Z^f und damit auch P^f für alle Fahrzeuge. Diese Mengen werden daher im Folgenden einfach nur Z bzw P genannt. Für die Länge einer Tour p wird die Notation $l_p = \sum_{a \in A} x_{pa} l_a$ verwendet. Außerdem sei $\lambda_p = \sum_{f \in F} \lambda_p^f$.

Die aggregierte Form des Problems lautet dann

$$\min \sum_{p \in Z} l_p \lambda_p \quad \text{s.t.} \tag{6.16}$$

$$\sum_{p \in Z} x_{pa} \lambda_p = 1 \qquad a \in A_R \tag{6.17}$$

$$\sum_{p \in Z} \lambda_p = |F| \tag{6.18}$$

$$\lambda_p \in \{0, 1\} \qquad p \in P \tag{6.19}$$

Auf Grund der Annahme, dass beliebig viele Fahrzeuge zur Verfügung stehen und wegen der Zulässigkeit „leerer Touren“, d.h. Touren in denen ein Fahrzeug im Depot verbleibt, kann die aggregierte Konvexitätsbedingung (6.18) vernachlässigt werden. Für das Masterproblem ergibt sich folgende Endversion

$$\min \sum_{p \in Z} c_p \lambda_p \quad \text{s.t.} \tag{6.20}$$

$$\sum_{p \in Z} x_{pa} \lambda_p = 1 \qquad a \in A_R \tag{6.21}$$

$$\lambda_p \in \{0, 1\} \qquad p \in Z \tag{6.22}$$

7 Branch-and-Price für den Straßenwinterdienst

In diesem Kapitel wird erläutert wie das Branch-and-Price-Verfahren für das in Kapitel 6 vorgestellte Straßenwinterdienstproblem angewendet werden kann. Das Branch-and-Price-Verfahren wurde in Kapitel 3 vorgestellt. Die Subprobleme, die sich durch Dekomposition des Straßenwinterdienstproblems ergeben sind Ressource-Constrained-Shortest-Path-Probleme. Diese können durch den in Kapitel 5 kurz vorgestellten Labeling-Algorithmus gelöst werden.

Es wird zuerst die Lösung des Originalproblems durch eine Savingsheurstik beschrieben. Für jede der Touren, die in der Heuristik gefunden werden, wird eine Spalte zum Masterproblem hinzugefügt. So erhält man das erste eingeschränkte Masterproblem und eine zulässige Lösung des Problems.

Es werden die selben Notationen wie in Kapitel 6 verwendet. Mit Masterproblem wird das Problem (6.20)-(6.22) bezeichnet.

7.1 Auffinden einer Startlösung: Savingsheuristik

Um die LP-Relaxierung des Masterproblems wie in Abschnitt 3.2 beschrieben zu lösen, braucht man eine Teilmenge der zulässigen Touren Z. Am Anfang des Verfahrens sind jedoch keine solchen Touren bekannt. Es gibt mehrere Möglichkeiten die benötigten Touren zu generieren. Eine davon ist die Heuristische Lösung des Originalproblems (6.5)-(6.13).
Die Lösung der Heuristik für das Originalproblem stellt sicher, dass die LP-Relaxierung des Masterproblems eine zulässige Lösung hat. Zugleich liefert diese Startlösung auch die dualen Informationen, die im Pricing verwendet werden, um neue Spalten zu generieren.

Das Verfahren kann auch ohne Startlösung durchgeführt werden, wenn das Farkas-Pricing (s. Abschnitt 3.5) verwendet wird.
Abschnitt 7.1 beschreibt ein Verfahren, dass eine ganzzahlige Startlösung für das Masterproblem (6.20)-(6.22) liefert.

7.1.1 Bilden der Distrikte

In der Praxis ist es üblich, das Straßennetz in kleinere Distrikte aufzuteilen, für deren Räumung jeweils eine Straßenmeisterei zuständig ist. Verfahren zur Bildung von Distrikten können bei Kandula und Wright [16], Muyldermans et al. [22] und Benavent et al. [3] nachgelesen werden.
Die nachfolgend gebildeten Distrikte werden nur zum Auffinden einer Startlösung verwendet. Die Zuständigkeitsbereiche der Straßenmeistereien ergeben sich in der Optimallösung aus den Touren, die jeweils mit einer der Straßenmeistereien verbunden sind.

Um die Distrikte zu bilden, müssen nur die Kanten, die das Räumen eines bestimmten Straßenabschnittes repräsentieren, einem Distrikt (Depot) zugeteilt werden. Alle anderen Kanten dienen nur als Verbindung. Die Aufgaben, die einem Depot zugeordnet werden, müssen dann in den Touren, die an diesem Depot beginnen und enden, erledigt werden.

Eine einfache Aufteilung in Distrikte erhält man, in dem man jede Kante in A_R dem Depot zuordnet dem sie am nähesten ist. Dazu müssen die kürzesten Wege von jedem Depot zu allen Kanten berechnet werden. Da später die kürzesten Wege zwischen allen Knoten benötigt werden, wird dazu der All-Pairs-Shortest-Paths-Algorithmus von Johnson [14] verwendet. Dieser gibt eine Entfernungsmatrix $H \in \mathbb{R}^{|V|\times|V|}$ zurück, die die Längen der kürzesten Wege zwischen den Knoten enthält.

Die Distrike werden nun folgendermaßen gebildet:

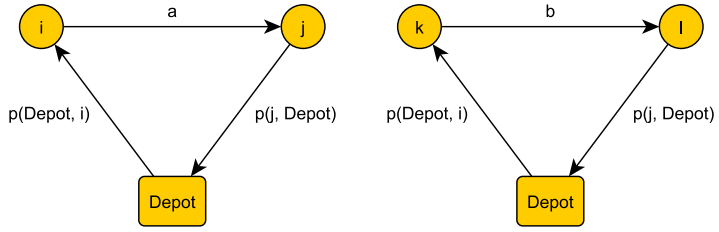
Algorithmus 1: Bilden der Distrikte

Sei g der Graph, der das Straßennetzwerk repräsentiert;
Berechne Johnsons All-Pairs-Shortest-Paths;
Sei H die Matrix, die die Längen der kürzesten Wege enthält;
Sei $Z_d \subset A_R$ die Menge der Kanten(Aufgaben) im Distrikt für Depot d;
for $d \in D$ **do**
  $Z_d = \emptyset$
end
for $a \in A_R$ **do**
  Sei i der Anfangsknoten von a;
  $d = \text{argmin } H[d, i]$;
  $Z_d = Z_d \cup \{a\}$;
end

7.1.2 Bilden der Touren

Es sollen nun für jedes Depot Touren gefunden werden, sodass alle Kanten eines Distrikts Z_d in einer der Touren enthalten sind, die an dem zugehörigen Depot starten und enden. Dabei soll die insgesamt gefahrene Strecke möglichst gering sein. Dazu wird hier ein Einsparkriterium verwendet, das dem von Clarke und Wright [5] für das Vehicle-Routing-Problem ähnelt.

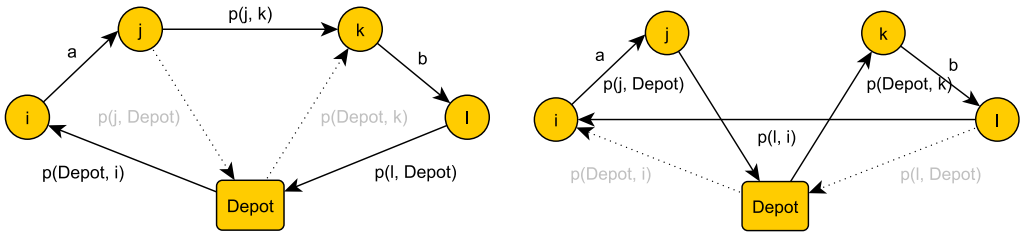
Zuerst wird für die Räumung jedes Straßenabschnittes in einem Distrikt eine eigene Tour gebildet. Für zwei Kanten a und b beginnt man mit den Touren aus Abb. 7.1. Hier bezeichnet $p(i, j)$ einen kürzesten Weg von Knoten i nach Knoten j.



(a) Tour für Kante a (b) Tour für Kante b

Abbildung 7.1: Touren Begin

Die Kanten a und b repräsentieren das Räumen der Straßenabschnitte (i, j) und (k, l). Die Idee besteht darin, beide Aufgaben (a und b) in einer gemeinsamen Tour zu erledigen. Dazu gibt es zwei Möglichkeiten. Entweder wird (i, j) vor (k, l) geräumt oder umgekehrt.



(a) a vor b (b) b vor a

Abbildung 7.2: Zusammenfügen der Touren

H ist die Entfernungsmatrix, d.h. $H\,[i]\,[j]$ ist die Entfernung zwischen Knoten i und j. Die Streckeneinsparungen lassen sich leicht berechnen. Für den in Abb. 7.2(a) dargestellten Fall ergeben sich die Einsparungen S_a folgendermaßen:

$$
\begin{aligned}
& H\,[\text{Depot}]\,[i] + l_a + H\,[j]\,[\text{Depot}] && \text{s. Abb. 7.1(a)} \\
+\ & H\,[\text{Depot}]\,[k] + l_b + H\,[l]\,[\text{Depot}] && \text{s. Abb. 7.1(b)} \\
-\ & H\,[\text{Depot}]\,[i] + l_a + H\,[i]\,[j] + l_b + H\,[l]\,[\text{Depot}] && \text{s. Abb. 7.2(a)} \\
=\ & H\,[\text{Depot}]\,[k] + H\,[j]\,[\text{Depot}] - H\,[j]\,[k] && (7.1) \\
=\ & S_a
\end{aligned}
$$

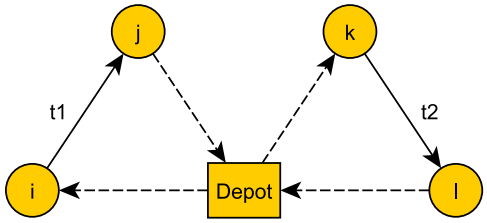
Für Abb. 7.2(b) ergeben sich die Einsparungen S_b

$$
\begin{aligned}
& H\,[\text{Depot}]\,[k] + l_b + H\,[l]\,[\text{Depot}] && \text{s. Abb. 7.1(b)} \\
+\ & H\,[\text{Depot}]\,[i] + l_a + H\,[j]\,[\text{Depot}] && \text{s. Abb. 7.1(a)} \\
-\ & H\,[\text{Depot}]\,[k] + l_b + H\,[l]\,[i] + l_a + H\,[j]\,[\text{Depot}] && \text{s. Abb. 7.2(b)} \\
=\ & H\,[\text{Depot}]\,[i] + H\,[l]\,[\text{Depot}] - H\,[l]\,[i] && (7.2) \\
=\ & S_b
\end{aligned}
$$

Ist S_a oder S_b positiv, lohnt es sich die beiden Touren aus Abb. 7.1 zusammenzulegen. Am Anfang werden nun für alle Kantenpaare im Distrikt Z_d (s. Algorithmus 1) die Einsparungen berechnet, wenn die zugehörigen Straßenabschnitte statt in zwei einzelnen Touren in einer gemeinsamen Tour geräumt werden.

Sei $a - b$ das Kantenpaar mit dem sich durch Zusammenlegung der Touren die größten Einsparunge erzielen lassen. Falls die Länge der zusammengefügten Tour kleiner als l_{max} ist, werden die Touren für a und b so zusammengefügt, dass a direkt vor b geräumt wird. Die neu entstandene Tour wird zu der Liste aller Touren hinzugefügt. Die beiden Einzeltouren werden von dieser Liste gelöscht.

Seien t_1 und t_2 Touren, in denen mehrere Kanten geräumt werden. Sei i der Anfangsknoten der ersten Straßenabschnitts, der in t_1 geräumt wird, und k der Anfangsknoten der ersten Kante, die in t_2 geräumt wird. Die Knoten j bzw. l seien die Endknoten der Straßenabschnitte, die als letztes in t_1 bzw. t_2 geräumt werden.



Die Einsparungen für das Tourenpaar (t_1, t_2) ergeben sich wie in (7.1) und für das Tourenpaar (t_2, t_1) wie in (7.2)

Algorithmus 2: Savingsheuristik

Rufe Algorithmus 1 auf und erhalte Distrikte Z_d für $d \in D$;
Sei H die Entfernungsmarix der kürzesten Wege zwischen allen Knoten;
for $d \in D$ **do**
    Sei T_d die Liste aller Touren im Distrikt von Depot d;
    for $a \in Z_d$ **do**
        Sei i der Anfangsknoten und j der Endknoten der Kante a;
        // Bilde für Kante a eine eingene Tour t
        $t = (a)$;
        $l(t) = H[d, i] + l_a + H[j, d]$; // Länge der Tour t
        $\text{beg}(t) = i$; // Anfangsknoten der ersten Kante in t
        $\text{end}(t) = j$; // Endknoten der letzten Kante in t
        $T = T \cup t$;
    end
    Sei S die Matrix der Einsparungen, wobei $S[t_1][t_2]$ die Einsparungen für Tourenpaar (t_1, t_2) sind;
    while $b = true$ **do**
        for $t_1 \in T$ **do**
            for $t_2 \in T$ **do**
                // Berechne Einsparungen s für Tourenpaar (t_1, t_2) nach (7.1)
                $s = H[d, \text{beg}(t_2)] + H[\text{end}(t_1), d] - H[\text{end}(t_1), \text{beg}(t_2)]$;
                $S[t_1][t_2] = s$;
            end
        end
        // Finde ein Tourenpaar, das die Ersparnis maximiert
        $(t_1, t_2) = \text{argmax}\, S[i][j]$;
        $s_{max} = \max S[t_1][t_2]$;
        // Falls es kein Tourenpaar mit positiven Einsparungen gibt, ist man fertig
        if $s_{max} < 0$ **then**
            b=false;
            break;
        end
        // Füge Touren t_1 und t_2 zu einer neuen Tour q zusammen
        Sei $p(\text{end}(t_1), \text{beg}(t_2))$ der kürzeste Weg um die beiden Touren zu verbinden;
        $q = (t_1, p(\text{end}(t_1), \text{beg}(t_2)), t_2)$;
        $l(q) = H[d, \text{beg}(t_1)] + l(t_1) + H[\text{end}(t_1), \text{beg}(t_2)] + l(t_2) + H[\text{end}(t_2), d]$;
        $\text{beg}(q) = \text{beg}(t_1)$;
        $\text{end}(q) = \text{end}(t_2)$;
        $T = T \setminus (t \cup p)$; // Löschen der beiden Touren t_1 und t_2
        $T = T \cup q$; // Hinzufügen der Tour q
    end
end

7.2 Pricing: Resource-Constraint-Shortest-Path

Die Aufgabe des Pricing-Problems besteht darin, neue zulässige Touren mit negativen reduzierten Kosten zu generieren oder nachzuweisen, dass die aktuelle Lösung der LP-Relaxierung des eingeschränkten Masterproblems optimal für die LP-Relaxierung des Masterproblems ist.

Die negativen reduzierten Kosten einer Tour $p \in Z$ mit zugehörigem Inzidenzvektor $x_p \in P$ sind gegeben durch

$$z_p = \sum_{a \in A} l_a x_{pa} - \sum_{a \in A_R} x_{pa} \pi_a \tag{7.3}$$

wobei π_a die Dualvariable zur Set-Partitioning-Nebenbedingung (6.21) für Kante a des Masterproblems ist.

Im Pricing-Problem soll also eine Tour p berechnet werden, die an einem Depot beginnt, am selben Depot endet, nicht länger als l_{max} ist und für die z_p echt kleiner Null ist. Außerdem sollen die Kanten $a \in A_R$ in jeder Tour höchstens einmal vorkommen. Eine solche Tour kann durch die Lösung eines Elementary-Resource-Constrained-Shortest-Path-Problems (ERCSPP) erzeugt werden. Die Gesamtlänge der Tour ist hier die einzige beschränkte Ressource. Die negativen reduzierten Kosten (7.3) sollen minimiert werden.

Neben der Länge werden jeder Kante $a \in A$ daher ihre Kosten c_a zugeordnet, wobei

$$c_a = \begin{cases} l_a & \text{für } a \in \bar{A_R} \\ l_a - \pi_a & \text{für } a \in A_R \end{cases}$$

ist.

Damit jede Tour auch die strukturelle Anforderung, dass jede Kante $a \in A_R$ nur einmal in einer Tour vorkommen darf, erfüllt, wird für jede dieser Kanten eine weitere Ressource benötigt, die angibt, ob die betreffende Kante schon in der Tour enthalten ist oder nicht. Eine Tour ist nur dann zulässig, wenn sie jede der Kanten $a \in A_R$ höchstens einmal enthält.

7.2.1 Resource-Extension-Funktionen (REF)

Die REF für die Länge und die reduzierten Kosten einer Tour beschreiben die Veränderung dieser beiden „Ressourcen“ entlang einer Kante a. Die Ressourcenvariablen für die Länge und die Kosten seien T^l und T^c. T^l_v bezeichnet die Länge der Tour vom Startknoten zum Knoten v und T^c_v sind die Kosten der Tour bis zum Knoten v.
Sei $g(a) = (i, j)$ das Paar von Anfangs- und Endknoten der Kante a. Gegeben seien die Länge T^l_i und die Kosten T^c_i der Tour bis zum Knoten i. Nachfolgend werden die REF für die Länge und die Kosten gegeben.

Funktion REFLänge(Ressourcenvariable T_i^l, Kante $a \in A$)

$f = T_i^l + l_a$;
return f;

Funktion REFKosten(Ressourcenvariable T_i^c, Kante $a \in A$)

$f = T_i^c + c_a$;
return f;

Die zulässigen Intervalle in einem Knoten $v \in V$ für die Länge einer Tour sind

$$\begin{aligned} [0,0] &\quad \text{für den Startknoten } v = d_s \text{ der Tour} \\ [0, l_{max}] &\quad \text{für alle } v \in V \setminus d_s \end{aligned}$$

um sicherzustellen, dass die Tour nicht länger als l_{max} sein kann.

Für die reduzierten Kosten gibt es keine Beschränkung. Es werden daher auch keine zulässigen Intervalle benötigt.

Für jede Kante $a \in A_R$ wird eine zusätzliche Ressource r_a eingeführt. Für jede dieser Ressourcen wird eine REF benötigt, die sicherstellt, dass die zugehörige Kante höchstens einmal in einer Tour vorkommen kann.

Sei $T_i^{r_e}$ die zu $e \in A_R$ gehörende Ressourcenvariable in einem Knoten $i \in V$. Wird eine Tour über eine Kante $a \in \delta^+(i)$ erweitert, ist die REF gegeben durch

Funktion REFe(Ressourcenvariable $T_i^{r_e}$, Kante a)

if $a = e$ **then** $f = T_i^{r_e} + 1$;
else $f = T_i^{r_e}$;
return f;

Die zulässigen Intervalle für die Ressourcenvariable T^{r_a} in den Knoten $v \in V$ sind

$$\begin{aligned} [0,0] &\quad \text{für den Startknoten } v = d_s \text{ der Tour} \\ [0,1] &\quad \text{für } v \in V \setminus d_s \end{aligned}$$

Falls also eine Kante $a \in A_R$ zweimal in einer Tour vorkäme, so wäre der Wert der zugehörigen Ressourcenvariable ebenfalls zwei und somit unzulässig in allen Knoten des Graphen.

Damit nicht alle zulässigen Touren durch den Graphen erzeugt werden, werden Touren, die nicht zu einer pareto-optimalen Tour erweitert werden können, mit Hilfe einer Dominanzfunktion verworfen.

7.2.2 Dominanzfunktion

Gegeben seien zwei Teiltouren p und q, die den selben Endknoten $v = v(p) = v(q)$ haben. $(T_v)^p$ und $(T_v)^q$ seien die Ressourcenvektoren der Touren im Knoten v. Die Tour p dominiert Tour q, falls es für jede zulässige Erweiterung von q eine zulässige Erweiterung von p gibt, sodass die Werte der Ressourcen in der Erweiterung von p immer kleiner oder gleich den Werten der Ressourcen in der Erweiterung von q sind. Sei $A_R(p)$ die Menge der Kanten $a \in A_R$, die in p enthalten sind. Eine Erweiterung für q kann nur dann immer zulässig für p sein, wenn $A_R(p) \subseteq A_R(q)$ gilt. Damit p also q dominieren kann, muss für jede Kante $a \in A_R$ gelten, dass der Wert der zugehörigen Ressourcenvariable für p kleiner gleich dem Wert für q ist. Das heißt

$$(T_v^{r_a})^p \leq (T_v^{r_a})^q \tag{7.4}$$

muss für alle $a \in A_R$ erfüllt sein.

Ist die Vorraussetzung 7.4 für alle $a \in A_R$ erfüllt, dominiert Tour p Tour q, falls

$$(T_v^l)^p \leq (T_v^l)^q$$

und

$$(T_v^c)^p \leq (T_v^c)^q$$

gilt.

7.3 Branching

Wurde mittels Spaltenerzeugung eine fraktionale Lösung für das Masterproblem (6.20)-(6.22) gefunden, soll die zulässige Menge des Problems so verzweigt werden, dass die fraktionale Lösung ausgeschlossen wird, alle zulässigen ganzzahligen Lösungen jedoch erhalten bleiben.

Die in Abschnitt 4.2.1 vorgestellte Branchingstrategie von Ryan und Foster kann für das Set-Partitioning-Masterproblem (6.20)-(6.22) angewendet werden. Verzweigen auf den Variablen des Originalproblems (6.5)-(6.5) ist nicht möglich, da wegen der Homogenität der Fahrzeuge ein aggregiertes Sub- und Masterproblem verwendet werden.

7.3.1 Ryan-Foster-Branchingregel

Um die Ryan-Foster-Branchingregel für das Masterproblem anwenden zu können, müssen im Fall einer fraktionalen LP-Lösung nach Satz 4.1 zwei Zeilen r und s gefunden werden,

für die (4.6) gilt. Im Fall des Masterproblems (6.20)-(6.22) entsprechen die Zeilen r und s zwei Aufgaben (Kanten) a_1 und a_2 des Graphen, wobei $a_1, a_2 \in A_R$.

Wie zwei passende Kanten gefunden werden können, kann aus dem Beweis zu Satz 4.1 geschlossen werden. Dazu betrachtet man die Variablen (Touren), die in der Lösung der LP-Relaxierung einen fraktionalen Wert annehmen. In einer solchen Tour p betrachtet man paarweise verschiedene Kanten a_1 und a_2. Gibt es eine andere Tour p', deren Variable auch einen fraktionalen Wert besitzt und die entweder a_1 oder a_2 aber nicht beide beinhaltet, dann sind a_1 und a_2 zwei Kanten, für die die Branchingentscheidungen formuliert werden können. Wie in Satz 4.1 gezeigt wurde, muss es zwei solche Kanten immer dann geben, wenn die Lösung der LP-Relaxierung fraktional ist. Sind die beiden Kanten gefunden, werden zwei neue Knoten im Branch-and-Bound-Baum erzeugt, Wobei im ersten Knoten nur Touren p zulässig sind, die entweder sowohl a_1 als auch a_2 enthalten oder weder a_1 noch a_2. Das heißt in einem Knoten sind nur Touren zulässig, für die

$$\begin{aligned} x_{pa_1} = x_{pa_2} = 1 \quad &\text{oder} \\ x_{pa_1} = x_{pa_2} = 0 \end{aligned}$$

gilt. Im zweiten Knoten sind nur Touren zulässig, in denen entweder a_1 oder a_2 oder keine der beiden Kanten enthalten sind. Es muss

$$\begin{aligned} x_{pa_1} = 0, x_{pa_2} = 1 \quad &\text{oder} \\ x_{pa_1} = 1, x_{pa_2} = 0 \quad &\text{oder} \\ x_{pa_1} = x_{pa_2} = 0 \end{aligned}$$

gelten.

Es gibt im Allgemeinen mehrere Kantenpaare, für die die Branchingentscheidungen formuliert werden können. Damit der Branch-and-Bound-Baum möglichst ausgeglichen ist, können die Kanten a_1 und a_2 so gewählt werden, dass sie möglichst viele Spalten, deren Variablen einen fraktionalen Wert in der aktuellen Lösung der LP-Relaxierung haben, trennen. Sei $n(a_1, a_2)$ die Anzahl der Touren, die sowohl a_1 als auch a_2 enthalten. $n(a_1)$ bzw. $n(a_2)$ seien die Anzahl der Touren, die a_1 bzw a_2 enthalten. Für jedes Kantenpaar (a_1, a_2), das sich zur Verzweigung des Problems eignen würde, werden $n(a_1, a_2)$, $n(a_1)$ und $n(a_2)$ berechnet. Es soll das Kantenpaar für die Verzweigung ausgewählt werden, für das

$$w(a_1, a_2) = |n(a_1, a_2) - (n(a_1) - n(a_1, a_2)) - (n(a_1) - n(a_1, a_2))|$$

minimal ist.

Algorithmus 3: Finden der Branching Kandidaten

Sei frak der Vektor, der alle fraktionalen Variablen der LP-Lösung enthält;
Sei n_{frak} die Anzahl der fraktionalen Variablen;
for $v = 0$ *bis* n_{frak} **do**
  Sei p die zu frak $[v]$ gehörende Tour;
  r_p sei der Vektor der alle Kanten $a \in A_R$ aus p enthält;
  n_p sei die Länge des Vektord r_p;
  for $i = 0$ *bis* n_p **do**
    $a_1 = r_p(i)$;
    $n(a_1) = n(a_1) + 1$;
    for $j = 0$ *bis* n_p **do**
      $a_2 = r_p(j)$;
      if $a_1 < a_2$ **then**
        $n(a_1, a_2) = n(a_1, a_2) + 1$;
      end
      else
        $n(a_2, a_1) = n(a_2, a_1) + 1$;
      end
    end
  end
end
for *alle Kantenpaare* (a_1, a_2) *mit* $a_1, a_2 \in A_R$ *und* $a_1 < a_2$ **do**
  $\overline{n_{a_1}} = n(a_1) - n(a_1, a_2)$; `// Anzahl Touren, die` a_1 `enthalten, nicht aber` a_2
  $\overline{n_{a_2}} = n(a_2) - n(a_1, a_2)$; `// Anzahl Touren, die` a_2 `enthalten, nicht aber` a_1
  $w(a_1, a_2) = |n(a_1, a_2) - \overline{n_{a_1}} - \overline{n_{a_2}}|$;
end
Wähle das Kantenpaar (a_1, a_2) mit dem kleinsten Wert $w(a_1, a_2)$ für das Ryan-Foster-Branching aus;

Eine andere Möglichkeit unter den Kandidaten-Paaren (a_1, a_2) ein Paar auszuwählen, ist den Wert

$$f(a_1, a_2) = \sum_{p: x_{pa_1}=1, x_{pa_2}=1} \lambda_p$$

zu vergleichen. Je größer der Wert dieser Summe ist, desto „besser“ ist das Kantenpaar. Denn ist der Wert nahe bei eins, dann ist es sehr wahrscheinlich, dass die zu den Kanten gehörenden Aufgaben in der selben Tour erledigt werden. Gibt es Paare mit $f(a_1, a_2) = 1$, wird festgelegt, dass diese beiden Aufgaben in einer Tour erledigt werden. Diese Festlegung schließt allerdings die aktuelle fraktionale Lösung nicht aus. Daher wird dann auf dem Kantenpaar mit dem größten fraktionalen Wert $f(a_1, a_2)$ verzweigt.

7.4 Berücksichtigung der Branchingentscheidungen im Pricing

Es muss nun sichergestellt werden, dass in jedem Knoten des Branch-and-Bound-Baumes die in diesem Knoten gültigen Branchingentscheidungen auch bei der Erzeugung neuer Spalten eingehalten werden. Im ERCSPP sind also nur noch Touren zulässig, die die Ryan-Foster-Branchingentscheidungen einhalten.

Wie in Abschnitt 5.2 erläutert, können Ryan-Foster-Branchingentscheidungen im RCSPP durch das Hinzufügen neuer Ressourcen und deren Resource-Extension-Funktionen berücksichtigt werden. Auch die Dominanzfunktion verändert sich, weil strukturelle Anforderungen an die Touren gestellt werden.

Sollen zwei Kanten a_1 und a_2 aus A_R in der selben Tour vorkommen, wird eine neue Ressource $it(a_1, a_2)$ eingeführt. Eine Tour ist unzulässig, wenn sie eine der beiden Kanten enthält, die andere aber nicht. Die REF und die zulässigen Intervalle für diese Ressource müssen diese Anforderung wiederspiegeln.

Gegeben der Wert der Ressourcenvariable $T_i^{it(a_1,a_2)}$ lautet die REF für eine Kante $a \in \delta^+(i)$

Funktion REFgleicheTour(Ressourcenvariable $T_i^{it(a_1,a_2)}$, Kante a)

---

if $a = a_1$ **then** $f = T_i^{it(a_1,a_2)} + 1$;
else if $a = a_2$ **then** $f = T_i^{it(a_1,a_2)} - 1$;
else $f = T_i^{it(a_1,a_2)}$;
return f

Die zulässigen Intervalle müssen nun so gewählt sein, dass keine unzulässige Tour entstehen kann. Das heißt, dass im Anfangs- und im Endknoten der Tour die Ressourcenvariable $T^{it(a_1,a_2)}$ den Wert Null annhemen muss.
Soll die Tour am Startdepotknoten d_s^1 beginnen und am zugehörigen Zieldepotknoten d_t^1 enden, ist das zulässige Intervall für $it(a_1, a_2)$ im Knoten i gegeben durch

$$\begin{aligned} [0,0] &\quad \text{für } i = d_s^1, d_t^1 \\ [-1,1] &\quad \text{für } i \in V \setminus \left\{d_s^1, d_t^1\right\} \end{aligned}$$

Sollen zwei Kanten a_1 und a_2 aus A_R nicht in der selben Tour vorkommen, wird eine Ressource $fbit(a_1, a_2)$ eingeführt. Ist eine der beiden Kanten a_1 und a_2 schon in einer Tour enthalten, ist die Erweiterung dieser Tour über die Kanten a_1 und a_2 unzulässig. Wieder müssen die REF und die zulässigen Intervalle passend zu diesen Anforderungen gewählt werden.

Gegeben der Wert $T_i^{fbit(a_1,a_2)}$ der Ressource $fbit(a_1, a_2)$ in einem Knoten i lautet die REF für Kanten $a \in \delta^+(i)$

Funktion REFverschiedeneTouren(Ressourcenvariable $T_i^{fbit(a_1,a_2)}$, Kante a)

if $a = a_1$ *und* $T_i^{fbit(a_1,a_2)} = 0$ **then** $f = T_i^{fbit(a_1,a_2)} + 1$;
else if $a = a_2$ *und* $T_i^{fbit(a_1,a_2)} = 0$ **then** $f = T_i^{fbit(a_1,a_2)} - 1$;
else if $a \in \{a_1, a_2\}$ *und* $T_i^{fbit(a_1,a_2)} \neq 0$ **then** $f = \infty$;
else $f = T_i^{fbit(a_1,a_2)}$;
return f

Ist eine Tour vom Startdepotknoten d_s^1 zum Zieldepotknoten d_t^1 gesucht, ist für einen Knoten i das zulässige Intervall für den Wert der Ressource $T_i^{fbit(a_1,a_2)}$ gegeben durch

$$\begin{aligned} [0,0] &\quad \text{für } i = d_s^1 \\ [-1,1] &\quad \text{für } i \in V \setminus \{d_s^1\}. \end{aligned}$$

Der folgende Algorithmus bekommt den Ressourcenvektor T_i einer Teiltour, mit Endknoten $i \in V$ und eine Kante $a \in \delta^+(i)$ übergeben. Er berechnet die Ressourcenwerte im Zielknoten j der Kante a und gibt „true“ zurück, falls die Ressourcenwerte in j in den zulässigen Intervallen liegen.

Algorithmus 4: Resource-Extension-Funktionen

Input : Ressourcenvektor T_i
Kante $a \in \delta^+(i)$ mit $g(a) = (i, j)$
Ressourcenvektor T_j
Menge same der Kantenpaare, die nur zusammen in einer Tour vorkommen dürfen
Menge differ der Kantenpaare, die nicht in der selben Tour vorkommen dürfen
Zielknoten der Tour d_t

```
// REF für die Länge und die Kosten
```

$T_j^l = \text{REFLänge}(T_i^l, a)$;
if $T_j^l > l_{max}$ **then return** *false*;
$T_j^c = \text{REFKosten}(T_i^c, a)$;

```
// REF für die Ressourcen der Kanten
```

$e \in A_R$

for $e \in A_R$ **do**
  $T_j^{r_e} = \text{REFe}(T_i^{r_e}, a)$;
  if $T_j^{r_e} > 1$ **then return** *false*
end

```
// REF für die Branchingentscheidungen: Kanten in gleicher Tour
```

for $i = 0$ *bis* $|\mathsf{same}|$ **do**
  a_1 und a_2 sind die Kanten der Branchingentscheidung $\mathsf{same}(i)$;
  $T_j^{it(a_1,a_2)} = \text{REFgleicheTour}(T_i^{it(a_1,a_2)}, a)$;
  if $j = d_t$ *und* $T_j^{it(a_1,a_2)} \neq 0$ **then return** *false*;
end

```
// REF für die Branchingentscheidungen: Kanten nicht in gleicher Tour
```

for $i = 0$ *bis* $|\mathsf{differ}|$ **do**
  a_1 und a_2 sind die Kanten der Branchingentscheidung $\mathsf{differ}(i)$;
  $T_j^{fbit(a_1,a_2)} = \text{REFverschiedeneTouren}(T_i^{fbit(a_1,a_2)}, a)$;
  if $T_j^{fbit(a_1,a_2)} \notin [-1, 1]$ **then return** *false*;
end
return *true*;

Die Anforderungen an die Struktur einer Tour, die durch die Branchingentscheidungen entstehen, ändern auch die Dominanzfunktion.

Gegeben seien zwei Touren p und q mit dem selben Endknoten v. Die Tour p kann die Tour q nur dominieren, falls jede zulässige Erweiterung der Tour q auch eine zulässige Erweiterung der Tour p ist. Müssen keine Branchingentscheidungen berücksichtigt werden, ist die Dominanzfunktion wie in Abschnitt 7.2.2 gegeben. Im Folgenden wird erläutert, wie sich die Dominanzfunktion verändert, wenn Ryan-Foster-Branchingentscheidungen berücksichtigt werden müssen.

Gegeben seien zwei Teiltouren p und q mit dem selben Endknoten $v = v(p) = v(q)$. Sei $(T_v)^p$

der Ressourcenvektor der Tour p und $(T_v)^q$ der Ressourcenvektor der Tour q im Knoten v. Tour p kann Tour q nur dominieren, falls für alle Ressourcen $it(a_1, a_2)$

$$(T_v^{it(a_1,a_2)})^p = (T_v^{it(a_1,a_2)})^q$$

gilt und die Bedingungen (7.4) für alle $a \in A_R$ erfüllt sind. Ist $(T_v^{it(a_1,a_2)})^p = (T_v^{it(a_1,a_2)})^q = 0$, enthält jede der Touren entweder keine der beiden Kanten a_1 oder a_2 oder beide. Enthält Tour p beide Kanten und Tour q keine von beiden, ist Bedingung (7.4) nicht erfüllt. Falls p keine der beiden Kanten a_1 und a_2 enthält und q beide enthält, sind für q nur Erweiterungen zulässig in denen keine der beiden Kanten vorkommt. Diese Erweiterungen sind immer auch für p zulässig. Falls beide Touren keine der Kanten a_1 und a_2 enthalten, sind für q Erweiterungen zulässig, die entweder keine der Kanten oder beide beinhalten. Die selben Erweiterungen sind auch für p zulässig.
Gilt $(T_v^{it(a_1,a_2)})^p = (T_v^{it(a_1,a_2)})^q = 1$, enthalten beide Touren die Kante a_1. Für q sind also nur Erweiterungen zulässig, die a_2 enthalten. Diese Erweiterungen sind auch zulässig für p.
Ist $(T_v^{it(a_1,a_2)})^p = (T_v^{it(a_1,a_2)})^q = -1$, enthalten beide Touren a_2. Es sind für beide Touren also nur Erweiterungen zulässig, die a_1 enthalten.

Außerdem muss für alle Ressourcen $fbit(a_1, a_2)$

$$(T_v^{fbit(a_1,a_2)})^p = (T_v^{fbit(a_1,a_2)})^q$$

oder

$$(T_v^{fbit(a_1,a_2)})^p = 0$$

gelten.

Gilt $(T_v^{fbit(a_1,a_2)})^p = (T_v^{fbit(a_1,a_2)})^q$, dann enthalten beide Touren weder a_1 noch a_2 oder beide enthalten die selbe Kante. Falls beide Touren keine der Kanten enthalten, sind für beide Touren Erweiterungen zulässig, die höchstens eine der Kanten a_1 oder a_2 einschließen. Enthalten beide Touren a_1 so sind für beide Touren nur Erweiterungen zulässig, die a_2 nicht umfassen. Enthalten beide a_2 sind sowohl für p als auch q nur Erweiterungen zulässig, die a_1 nicht enthalten.

Gilt $(T_v^{fbit(a_1,a_2)})^p = 0$ und $(T_v^{fbit(a_1,a_2)})^q \neq 0$, so sind für q nur Erweiterungen zulässig, die keine der beiden Kanten a_1 oder a_2 enthalten. Diese Touren sind auch zulässig für p.

Gegeben seien also zwei Teiltouren q_1 und q_2, die den selben Endknoten $v = v(q_1) = v(q_2)$ haben. T_v^1 und T_v^2 seien die zugehörigen Ressourcenvektoren im Knoten v. Die Dominanzfunktion bekommt die beiden Ressourcenvektoren übergeben und gibt genau dann „true“ zurück, wenn q_1 q_2 dominiert.

Funktion Dominanzfunktion(T_v^1, T_v^2)

for $a \in A_R$ **do**
  if $(T_v^a)^1 > (T_v^a)^2$ **then**
    return *false*;
  end
end
for *alle Paare* (a_1, a_2)*, die in der selben Tour enthalten sein sollen* **do**
  if $(T_v^{it(a_1,a_2)})^1 \neq (T_v^{it(a_1,a_2)})^2$ **then**
    return *false*;
  end
end
for *alle Paare* (a_1, a_2)*, die nicht in der selben Tour enthalten sein sollen* **do**
  if $(T_v^{fbit(a_1,a_2)})^1 \neq (T_v^{fbit(a_1,a_2)})^2$ *und* $(T_v^{fbit(a_1,a_2)})^1 \neq 0$ **then**
    return *false*;
  end
end
if $(T_v^l)^1 \leq (T_v^l)^2$ *und* $(T_v^c)^1 \leq (T_v^c)^2$ **then**
  return *true*;
end

7.5 Implementierung des Pricings

Im Pricing werden für jede Straßenmeisterei Touren mit negativen reduzierten Kosten gesucht. Dazu wird für jede Straßenmeisterei ein ERCSPP gelöst, mit dem zu dieser Straßenmeisterei gehörendem Start- und Zieldepotknoten. Das ERCSPP erzeugt alle pareto-optimalen (undominierten) Touren, die an dieser Straßenmeisterei beginnen und enden. Für alle pareto-optimalen Touren mit negativen reduzierten Kosten, die im ERCSPP identifiziert wurden, wird eine Spalte zum Masterproblem hinzugefügt.

Für die Lösung der ERCSPP wurde die, für die Boost-Graphen-Bibliothek implementierte, r_c_shortest_paths-Funktion verwendet. Die Funktion ist die Implementierung eines Label-Setting-Algorithmus, wie er in Abschnitt 5.3 kurz beschrieben wurde. Der Labeling-Algorithmus musste daher nicht implementiert werden. Der Boost-Funktion müssen die betrachteten Ressourcen, die Resource-Extension-Funktionen, die Dominanzfunktion sowie der Start- und Zielknoten der Tour übergeben werden. Für Details zur Boost-Graphen-Bibliothek siehe `http://www.boost.org/doc/libs/1_42_0/libs/graph/doc/table_of_contents.html`,

Algorithmus 5: Pricing

```
Sei Touren die Liste aller Touren;
Sei D die Anzahl der Straßenmeistereien;
for i = 0 bis D do
    d_s und d_t sind die zur Straßenmeisterei gehörenden Depotknoten;
    Führe die r_c_shortest_paths-Funktion von Boost aus;
    optsol sei der von der Funktion zurückgegebene Vektor der pareto-optimalen
    Lösungen, wobei jedes Element von optsol ein Vektor von Kanten ist.;
    n_optsol sei die Anzahl der pareto-optimalen Touren;
    for j = 0 bis n_optsol do
        Sei p = optsol(i);
        T = T(p) ist der Ressourcenvektor der Tour p im Endknoten d_t der Tour;
        if T^c < 0 then
            Touren = Touren ∪p;
            Sei m die Anzahl der Kanten in Tour p;
            Sei x_p ein Vektor der Länge |A_R|;
            x_p = 0;
            for k = 0 bis m do
                a = p(k) sei die k − te Kante in p;
                if a ∈ A_R then
                    x_pa = 1;
                end
            end
            Füge den Vektor x_p als neue Spalte mit Kosten T^l zum Masterproblem hinzu;
        end
    end
end
```

7.6 Relaxiertes Pricing-Problem

Das RCSPP ist $\mathcal{NP}$-schwer [8]. Müssen die Touren nicht einfach sein, lässt sich das RCSPP in pseudo-polynomialer Zeit lösen [13]. Es ist daher möglicherweise attraktiver das RCSPP statt des ERCSPP im Pricing zu lösen.
Wird im Fall des Straßenwinterdienstproblems das RCSPP statt dem ERCSPP im Pricing gelöst, können unzulässige Touren erzeugt werden, die Kanten $a \in A_R$ mehr als einmal enthalten. Enthält eine Tour p eine Kante a k-mal, hat der Inzidenzvektor x_p den Eintrag $x_{pa} = k$. Die Set-Partitioning-Bedingungen (6.21) des Masterproblems garantieren, dass keine der Touren, deren Inzidenzvektor für eine Kante $a \in A_R$ einen Eintrag $x_{pa} > 1$ enthält, in einer ganzzahligen Lösung des Masterproblems (6.20)-(6.22) vorkommen kann.

Die Branchingregel von Ryan und Foster kann bei Verwendung des RCSPP statt des ERCSPP im Pricing-Problem nicht ohne weiteres angewendet werde. Denn falls eine der Varia-

blen, die zu einer unzulässigen Tour gehören, einen positiven Wert in der LP-Relaxierung des Masterproblems hat, ist es möglich, dass kein geeignetes Katenpaar für das Ryan-Foster-Branching gefunden werden kann. Hat eine Tour p z.B. nur für eine Kante $a \in A_R$ einen positiven Eintrag $x_{pa} = 2$, und die zugehörige Variable sei $\lambda_p = 0,5$, gibt es keine andere Spalte in der Lösung der LP-Relaxierung, die ebenfalls die Kante a überdeckt. Die Ryan-Foster-Branchingregel kann also nur angewendet werden, falls im Pricing tatsächlich das Elementary-Resource-Constrained-Shortest-Path-Problem gelöst wird.

Eine Möglichkeit zur Handhabung dieses Problems besteht darin, im Falle einer fraktionalen Lösung der LP-Relaxierung des eingeschränkten Masterproblems zu prüfen, ob in dieser Lösung die Variable einer unzulässigen Tour einen positiven Wert hat. Ist dies der Fall, kann diese Variable aus dem Masterproblem gelöscht werden. Damit diese Tour nach der nächsten Pricing-Runde nicht wieder zum Masterproblem hinzugefügt wird, muss für jede Tour, die dem Masterproblem hinzugefügt wird, überprüft werden, ob diese Tour auch wirklich „neu“ ist.

Funktion neueTour(Tour p)

```
Sei Touren die Liste aller Touren und n_T die Anzahl aller Touren;
for i = 0 bis n_T do
    if p = Touren(i) then
        return false;
    end
end
return true;
```

Hat keine zu einer unzulässigen Tour gehörende Variable einen positiven Wert in der Lösung der LP-Relaxierung des eingeschränkten Masterproblems, kann das Ryan-Foster-Branching aus Abschnitt 7.3 durchgeführt werden. Andernfalls muss der Branchingalgorithmus 3 erweitert werden. Es muss für alle Variablen, die einen positiven Wert in der Lösung der LP-Relaxierung des Masterproblems haben, überprüft werden, ob sie zu einer unzulässigen Tour gehören. Ist das der Fall werden diese Variablen aus dem Masterproblem gelöscht.

Algorithmus 6: Branching

Sei frak der Vektor, der alle fraktionalen Variablen der LP-Lösung enthält;
Sei n_{frak} die Anzahl der fraktionalen Variablen;
r = 0;
for $i = 0$ *bis* n_{frak} **do**
    p sei die zu frak $[i]$ gehörende Tour;
    x_p ist der zu p gehörende Inzidenzvektor;
    for $a \in A_R$ **do**
        if $x_{pa} > 1$ **then**
            Lösche Variable frak $[i]$ aus dem Masterproblem;
            $r = 1$
        end
    end
end
`// falls keine unzulässige Tour eine Variable mit Wert größer null in der Lösung hatte`
if $r = 0$ **then**
    Rufe Algorithmus 3 auf;
end

8 Ein Beispiel

Der Testgraph besteht aus 12 Knoten und 50 Kanten, von denen 26 Kanten das Räumen von Spuren und 24 das Durchfahren der Straßenabschnitte repräsentieren. Der Testgraph enthält zwei Depots, die durch jeweils zwei Knoten repräsentiert werden, einem Startdepotknoten und einem Zieldepotknoten. Der Testgraph ist in Abb. 8.1 dargestellt.

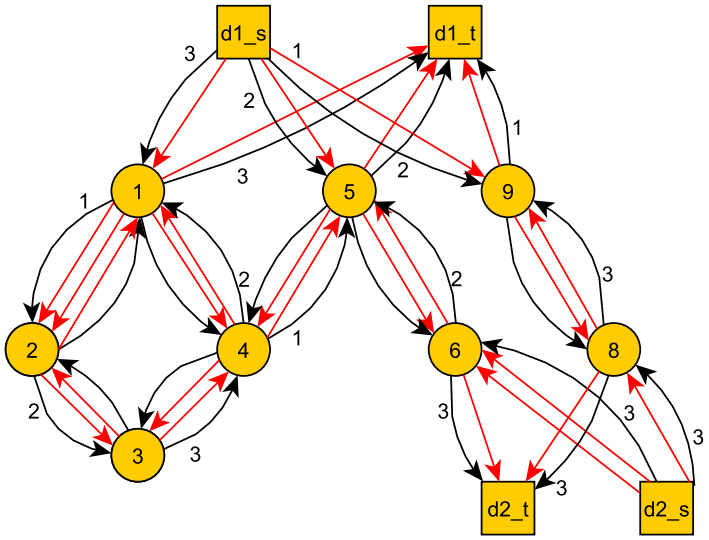


Abbildung 8.1: Testgraph

Die roten Kanten repräsentieren die Räumung einer Spur. Die schwarzen Kanten repräsentieren das Durchfahren eines Straßenabschnitts. Alle Kanten zwischen zwei Knoten haben die selbe Länge. Gesucht werden Touren für den Testgraphen, sodass jede rote Kante in genau einer Tour enthalten ist und die Summe der Kantengewichte kleiner als 15 ist.

8.1 Ergebnis der Savingsheuristik

Für das Testproblem findet die Savingsheuristik neun Touren, die die roten Kanten in Abb. 8.1 genau einmal überdecken. Die Gesamtlänge aller Touren beträgt 76.

In Abb. 8.2 sind die neun Fahrzeugtouren, die mit Hilfe der Savingsheuristik gefunden wurden, abgebildet. Die gestrichelten Linien stehen für das einfache Durchfahren eines Straßenabschnitts. Die durchgezogenen Linien stehen für das Räumen einer Spur.

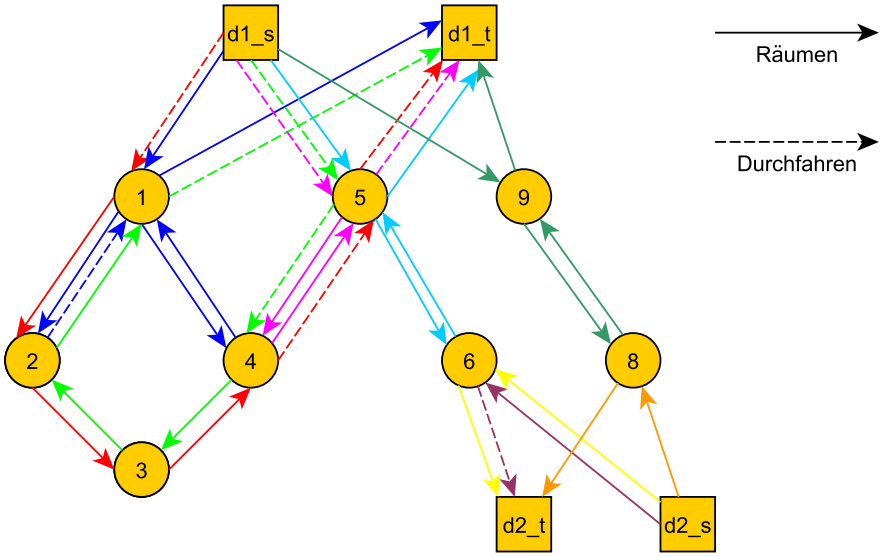


Abbildung 8.2: Ergebnis Savigsheuristik

Das Ergebnis kann durch einfache offensichtliche Schritte, wie z.B. das Austauschen von Kanten zwischen den Touren, verbessert werden. Da das Ergebnis der Heuristik jedoch nur als Startlösung verwendet wird, wird das Branch-and-Price-Verfahren mit dieser Lösung gestartet.

8.2 Ergebnis Branch-and-Price

Mit Hilfe des Branch-and-Price-Verfahrens wird dann eine Optimallösung für das Problem über dem Beispielgraphen gefunden. Das Problem wurde mit SCIP gelöst, wobei als LP-Löser CPLEX verwendet wurde. Infomationen zu SCIP findet man unter
`http://scip.zib.de/` .

Der optimale Zielfunktionswert für das Testproblem beträgt 66. Die Optimallösung ist demnach $13,16\%$ besser als die Lösung der Savingsheuristik. Es werden außerdem in der Lösung der Savingsheuristik zwei Fahrzeuge mehr benötigt als in der Optimallösung.

Im Vergleich zum Ergebnis der Savingsheuristik, wurden die Leerfahrten, das heißt das einfache Durchfahren von Straßenabschnitten, deutlich reduziert. Im Ergebnis der Savingsheuristik waren noch 14 solcher Kanten enthalten. Die Touren der Optimallösung weisen nur drei solcher Kanten auf.
Leerfahrten werden in der Optimallösung durch die Differenz von Eingangsgrad und Ausgangangsgrad in Knoten bedingt. Der Testgraph enthält z.B. zwei Kanten von 1 nach 2 die das Räumen von Spuren repräsentieren aber nur eine solche Kante von 2 nach 1. Für diejenigen Knoten, die in einer Tour enthalten sind muss jedoch immer Eingangsgrad gleich Ausgangsgrad gelten (s. (6.8)). Die Lösung des Problems müssen also mindestens so viele

Kanten, die das Durchfahren eines Straßenabschnittes repräsentieren, enthalten, dass für jeden Knoten des Graphen Eingangsgrad gleich Ausgangsgrad gilt.

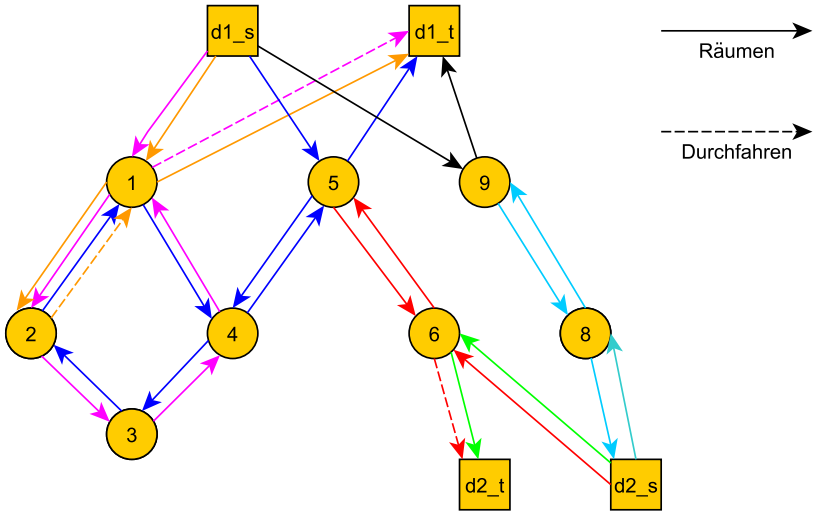


Abbildung 8.3: Ergebnis Branch-and-Price

Wird im Pricing-Problem das Elementary-Resource-Constrained-Shortest-Paths-Problem gelöst, braucht das Programm $1,93$ Sekunden, um ein optimale Lösung zu finden. Davon werden $1,51$ Sekunden für die Lösung der Pricing-Probleme und $0,16$ Sekunden für die Lösung der eingeschränkten Masterprobleme benötigt. Die meiste Zeit wird demnach für das Pricing verwendet. Insgesamt werden im Pricing 1350 Spalten erzeugt. Die Optimallösung wurde im 23ten Branch-and-Bound-Knoten in Tiefe vier nach $0,47$ Sekunden gefunden. Für die Lösung des Problems wurden insgesamt 111 Knoten im Branch-and-Bound-Baum gelöst.

Abb. 8.4 zeigt die Entwicklung der oberen und unteren Schranken für den Zielfunktionswert der LP-Relaxierung des Masterproblems im Wurzelknoten des Branch-and-Bound-Baums. Als obere Schranke für $\sum_{p \in Z} \lambda_p$ wird die in der Lösung der Savingsheuristik benötigte Anzahl der Fahrzeuge, also 11, gesetzt.

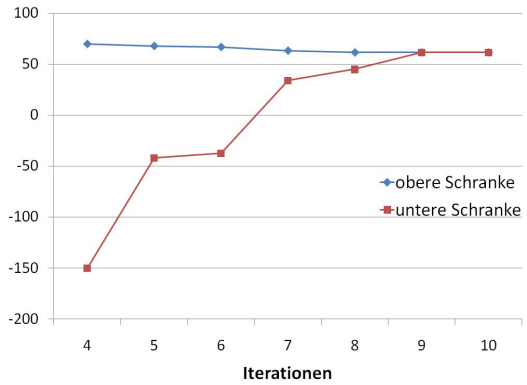


Abbildung 8.4: Entwicklung der oberen und unteren Schranken im Wurzelknoten

Löst man statt dem ERCSPP nur das Ressource-Constrained-Shortest-Path-Problem und verwendet das abgeänderte Branching aus Algorithmus 6 erhält man bereits nach $0,09$ Sekunden eine optimale Lösung des Problems. Es müssen insgesamt nur acht Branch-and-Bound-Knoten gelöst werden, um die Optimallösung zu finden und Optimalität nachzuweisen.
Die Entwicklung der oberen und unteren Schranken für die LP-Relaxierung ist Abb. 8.5 abgebildet. Die Schwankungen in der Entwicklung der unteren Schranke sind deutlich zu erkennen. Obere und untere Schranke entsprechen sich mehrmals bevor gebrancht wird. In diesen Fällen enthielt die optimale Lösung der LP-Relaxierung unzulässige Touren, die dann aus dem Masterproblem gelöscht wurden.

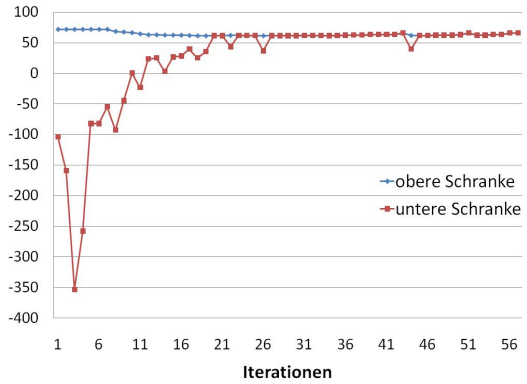


Abbildung 8.5: Entwicklung der oberen und unteren Schranken im Wurzelknoten

Wird das relaxierte Pricing-Verfahren verwendet, müssen also wesentlich mehr Iterationen durchgeführt werden, um eine optimale Lösung für die LP-Relaxierung des Masterproblems zu finden. Das Pricing-Problem ist jedoch wesentlich schneller lösbar als das Elementary-Ressource-Constrained-Shortest-Path-Problem. Die beste untere Schranke, die im Wurzelknoten des Branch-and-Bound-Baums gefunden wird entspricht bei Verwendung des relaxierten Pricing-Problems der Optimallösung. Bei Verwendung des des ERCSPP im Pricing

liegt die größte untere Schranke im Wurzelknoten des Branch-and-Bound-Baums bei 61,5 und ist damit deutlich geringer.

9 Lösen des Straßenwinterdienstproblems

Bis zur Fertigstellung dieser Arbeit war es nicht möglich das Straßenwinterdienstproblem für den Graphen, der das Straßennetz in den Landkreisen Lahn-Dill und Limburg-Weilburg repäsentiert, optimal zu lösen. Der Graph hat 3215 Kanten, von denen 1465 das Räumen von Spuren repräsentieren und 635 Knoten. Die Touren sollen von vier Depots aus geplant werden, die jeweils durch zwei Knoten im Graph repräsentiert werden.

9.1 Ergebnis der Savingsheuristik

Für das Straßennetz findet die Savingsheuristik 48 Fahrzeugtouren. 13 dieser Touren beginnen und enden in der Straßenmeisterei Dillenburg, 16 an der Straßenmeisterei Solms, sieben an der Straßenmeisteri Brechen und 12 an der Straßenmeisterei Weilburg. Auf diesen 48 Touren werden insgesamt 3747,05 km Strecke zurück gelegt.

Die insgesamt zu räumende Strecke im kompletten Straßennetz beträgt 2647,18 km. Dieser Wert ist eine untere Schranke für die von allen Fahrzeugen zusammen zu fahrende Strecke zur Räumung des Straßennetzes. Das Ergebnis der Savingsheuristik ist demnach höchstens 29,35% schlechter als die Optimallösung.

Dieses Ergebnis stellt auf jeden Fall sicher, dass das Problem zulässige Lösungen besitzt. Es wäre jedoch wünschenswert dieses Ergebnis weiter zu verbessern. Der kritische Punkt bei der Lösung des Problems ist die Lösung des Pricing-Problems.

9.2 Branch-and-Price-Heuristik

Um die Lösung zu beschleunigen, kann eine Branch-and-Price-Heuristik verwendet werden. In einer solchen Heuristik werden ab einer gewissen Knotentiefe im Branch-and-Bound-Baum keine neuen Spalten mehr erzeugt. In vielen Anwendungen werden nur im Wurzelknoten des Branch-and-Bound-Baumes Spalten erzeugt und dann auf diesen Spalten verzweigt. Durch ein solches Vorgehen ist das Auffinden einer Optimallösung oder überhaupt einer zulässigen Lösung allerdings nicht garantiert.

Um den sogenannten „tailing-off“-Effekt zu verhindern, kann gebrancht werden bevor eine optimale Lösung für die LP-Relaxierung des Masterproblems gefunden wird. Dazu kann man die oberen und unteren Schranken heranziehen, die man während der Spaltenerzeugung erhält (s.S. 18). Ist die Abweichung zwischen oberer und unterer Schranke sehr klein, wird gebrancht.

Da hier eine zulässige Lösung aus der Savingsheuristik bekannt ist, kann das Branch-and-Price-Verfahren verwendet werden, um diese Lösung zu verbessern. Wie gut die Lösung

der Savingsheuristik ist, kann anhand der unteren Schranken, die während des Branch-and-Price-Verfahrens bestimmt werden (s. S. 18), abgeschätzt werden.
In jedem Pricing-Schritt wird für jedes Depot ein RCSPP gelöst. Dabei werden üblicherweise sehr viele zulässige Touren mit negativen reduzierten Kosten gefunden. Werden für alle Touren mit negativen reduzierten Kosten Variablen und Spalten zum eingeschränkten Masterproblem hinzugefügt, ist es möglich, dass die Zeit, die benötigt wird um das eingeschränkte Masterproblem zu lösen, mit jeder Iteration erheblich zunimmt. Unter Umständen ist es daher empfehlenswert nur für einen Teil der gefundenen Touren mit negativen reduzierten Kosten Spalten zum Masterproblem hinzuzufügen.

Es wurde versucht das Straßenwinterdienstproblem für das Straßennetz der Landkreise Lahn-Dill und Limburg-Weilburg mit einer solchen Heuristik zu lösen. Es wurden nur bis zu Knoten in Tiefe drei des Branch-and-Bound-Baumes Touren erzeugt. Außerdem wurden pro Iteration und Depot maximal 50 Spalten (Touren) mit negativen reduzierten Kosten zum Masterproblem hinzugefügt. Insgesamt wird das Masterproblem daher pro Iteration um höchstens 200 Spalten erweitert. Als obere Schranke für $\sum_{p \in Z} \lambda_p$ kann die Anzahl der Fahrzeuge in der Lösung der Savingsheuristik verwendet werden. Damit gelten für den Zielfunktionswert der LP-Relaxierung des Masterproblems folgende Schranken:

$$z_{RMP} + 48 * c^* \leq z^*_{LMP} \leq z_{RMP}$$

Die untere Schranke kann verwendet werden, um die Güte der Lösung der Savingsheuristik und anderen ganzzahligen Lösungen des Problems zu bestimmen.

Wie für das Beispielproblem aus Kapitel 8 wird auch für die Lösung des Straßenwinterdienstproblems in den Kreisen Lahn-Dill und Limburg-Weilburg SCIP in Kombination mit CPLEX als Löser verwendet. Zum Zeitpunkt der Abgabe lag noch kein Ergebnis vor.

Abb. 9.1 zeigt die für die Lösung der Pricing-Probleme benötigte Zeit pro Iteration.

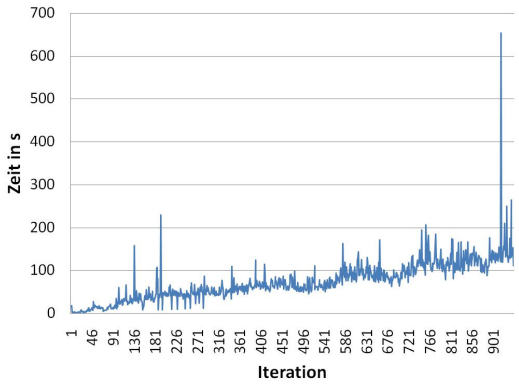
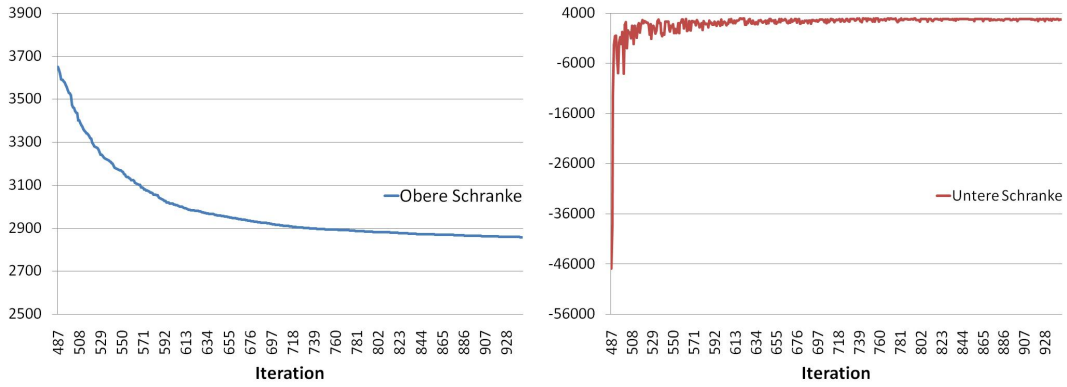


Abbildung 9.1: Zeit für das RCSPP Pricing

In Abb. 9.2 ist die Entwicklung der oberen und unteren Schranke für die LP-Relaxierung des Masterproblems im Wurzelknoten des Branch-and-Bound-Baums zu sehen.



(a) Obere Schranke

(b) Untere Schranke

Abbildung 9.2: Entwicklung der oberen und unteren Schranke im Wurzelknoten

In Abb. 9.2 ist der „tailing-off"-Effekt des Verfahrens zu sehen. In den ersten Iterationen ist der Fortschritt in der oberen Schranke groß. Je näher man jedoch dem Optimum kommt, desto kleiner ist der Fortschritt. Dieses Problem und mögliche Verfahren zur Minderung des Effekts werden in [20] vorgestellt. Die obere Schranke für den Zielfunktionswert ist monoton fallend, wie man in Abb. 9.2 gut erkennen kann. Die untere Schranke dagegen ist nicht monoton steigend sondern schwankt stark. Diese Eigenschaft der unteren Schranke wird als Problem angesehen [20]. Um die Schwankungen in der Enwicklung der unteren Schranke zu verhindern, können Stabilisierungsverfahren verwendet werden. Ein mögliches Verfahren wird von Lübbecke und Desrosiers [20] erläutert.

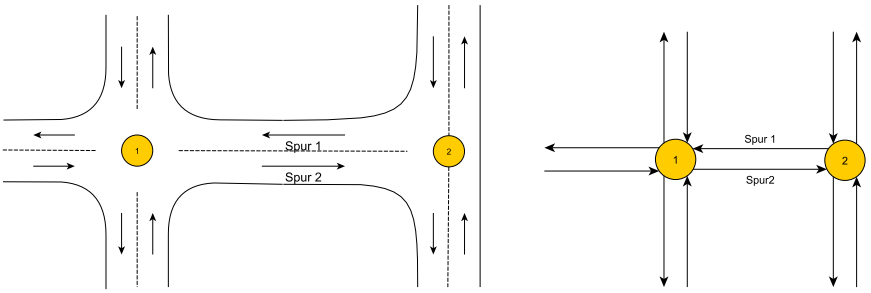
Der kritische Punkt bei der Lösung des Problems ist die Lösung des Pricing-Problems. Letchford und Oukil [18] haben für das CARP eine Formulierung des Pricing-Problems vorgeschlagen, die ausnutzt, dass die zugrunde liegenden Graphen häufig dünn (engl. sparse) sind. Möglicherweise ist die Lösung des Pricing-Problems als MIP eine besserer Lösungsansatz, als die Lösung des RCSPP durch dynamische Programmierung. Im folgenden Abschnitt wird eine alternative Formulierung für das Straßenwinterdienstproblem vorgestellt. Möglicherweise kann das durch Dekomposition dieser Formulierung entstehende Pricing-Problem schneller gelöst werden.

9.3 Alternative Formulierung

Die Formulierung des Straßenwinterdienstproblems aus Kapitel 6 ist nicht kompakt, weil sie exponentiel viele Nebenbedingungen besitzt. Perrier et al. [26] haben eine kompakte Formulierung für das Straßenwinterdienstproblem vorgeschlagen. Möglicherweise kann durch Dekomposition dieser Formulierung eine MIP-Formulierung für das Pricing-Problems gefunden werden, die durch Standard Lösungsverfahren für ganzzahlige Programme schneller

gelöst werden kann, als der Labeling-Algorithmus für das Resource-Constrained-Shortest-Path-Problem.

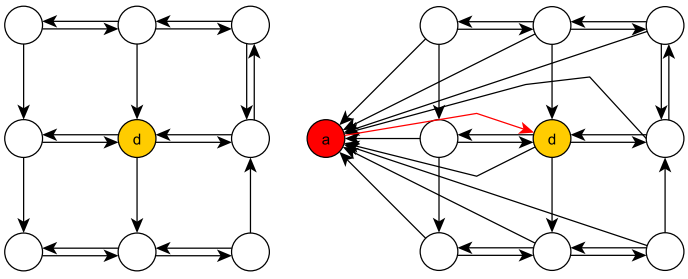
Es wird ein Graph $G(V, A, g)$ für das Straßennetz konstruiert, der für jede Richtungsspur genau eine gerichtete Kante enthält. Die Menge $A_R \subset A$ sei die Teilmenge der Kanten, die geräumt werden müssen.



(a) Kreuzung (b) Kreuzung als Graph

Abbildung 9.3: Kreuzung im Graph

Es wird ein zusätzlicher Knoten v_a und Kanten $A_1 = \{a : g(a) = (v_a, v_d), d \in D\}$ sowie Kanten $A_2 = \{a : g(a) = (v_i, v_a), v_i \in V\}$ zu diesem Graphen hinzugefügt. D sei die Menge der Depots und $V_D = \{v_d : d \in D\} \subset V$ die Menge der Depotknoten in G.
Der resultierende Graph $G' = G(V \cup v_a, A \cup A_1 \cup A_2, g)$ wird in Abb. 9.4 veranschaulicht.



(a) Graph G (b) Graph G'

Abbildung 9.4: Konstruktion von G' aus G

Die Menge $A' = A \cup A_1 \cup A_2$ ist die Menge aller Kanten in G'. Für jede Kante $a \in A'$ und jedes Fahrzeug $f \in F$ gibt die Variable y_{af} an, wie oft Fahrzeug f Kante a durchfährt (räumen oder durchfahren). Die Variable w_{af} repräsentiert den Fluss auf Kante a für Fahrzeug f. Für jede Kante $a \in A_R$ und jedes Fahrzeug $f \in F$ gibt die Variable x_{af} an, ob Kante a

von Fahrzeug f geräumt wird.

Das Straßenwinterdienst kann folgendermaßen formuliert werden:

$$\min \sum_{f \in F} \sum_{(a) \in A} y_{af} l_a \qquad \text{s.t.} \tag{9.1}$$

$$\sum_{f \in F} x_{af} = 1 \qquad a \in A_R \tag{9.2}$$

$$\sum_{a \in \delta^-(v)} y_{af} = \sum_{a \in \delta^+(v)} y_{af} \qquad v \in V \cup \{v_a\}, f \in F \tag{9.3}$$

$$\sum_{a \in A} y_{af} l_a \leq l_{max} \qquad f \in F \tag{9.4}$$

$$y_{af} \geq x_{af} \qquad a \in A_R, f \in F) \tag{9.5}$$

$$\sum_{a \in \delta^-(v)} w_{af} = \sum_{a \in \delta^+(v)} w_{af} \qquad v \in V \cup \{v_a\}, f \in F \tag{9.6}$$

$$y_{af} \leq w_{af} \leq |A|\, y_{af} \qquad a \in A \cup A_1, f \in F \tag{9.7}$$

$$y_{af} \leq w_{\bar{a}f} \qquad a \in A\ (g(a) = (i,j)), \bar{a} = (i, v_a), f \in F \tag{9.8}$$

$$\sum_{a \in A_2} y_{af} = 1 \qquad f \in F \tag{9.9}$$

$$x_{af} \in \{0,1\} \qquad a \in A_R, f \in F \tag{9.10}$$

$$y_{af} \geq 0 \text{ integer} \qquad a \in A', f \in F \tag{9.11}$$

$$w_{af} \geq 0 \qquad a \in A', f \in F \tag{9.12}$$

Die Zielfunktion (9.1) minimiert die insgesamt gefahrenen Strecke.
Durch die Bedingungen (9.2) ist sichergestellt, dass alle Kanten bedient werden. Die Bedingungen (9.3) garantieren, dass für jedes Fahrzeug und jeden Knoten Eingangsgrad gleich Ausgangsgrad ist. Dass keine der Fahrzeugtouren länger als l_{max} ist, wird durch die Nebenbedingungen (9.4) erreicht. Bedingungen (9.5) sagen, dass jede Kante, die geräumt wird, auch durchfahren wird. Die Flusserhaltung in jedem Knoten und für jedes Fahrzeug ist durch die Bedingungen (9.6) gewährleistet. Die Nebenbedingungen (9.7) stellen sicher, dass der Fluss auf einer Kante genau dann positiv ist, wenn sie durchfahren wird. Die Nebenbedingungen (9.8) garantieren, dass alle Fahrzeugtouren zusammenhängend sind (Beweis siehe [26]). Dass jede Fahrzeugtour mit genau einem Depot verbunden ist, wird durch die Bedingungen (9.9) erreicht.

Diese Formulierung ist wieder nach Fahrzeugen dekomponierbar. Das Masterproblem entspricht dem Masterproblem (6.20)-(6.22). Da immer noch davon ausgegangen wird, dass alle Fahrzeuge identisch sind, können die Subprobleme aggregiert werden. Im Subproblem wird versucht eine Spalte für das Masterproblem mit negativen reduzierten Kosten zu finden. Für jede Kante $a \in A_R$ gibt es eine Nebenbedingung im Masterproblem und eine

zugehörige Dualvariable π_a. Das Subproblem kann hier als MIP formuliert werden.

$$\min \sum_{(a)\in A} y_a l_a - \sum_{a\in A_R} \pi_a x_a \qquad \text{s.t.} \tag{9.13}$$

$$\sum_{a\in\delta^-(v)} y_a = \sum_{a\in\delta^+(v)} y_a \qquad v \in V \cup \{v_a\} \tag{9.14}$$

$$\sum_{a\in A} y_a l_a \leq l_{max} \tag{9.15}$$

$$y_a \geq x_a \qquad a \in A_R \tag{9.16}$$

$$\sum_{a\in\delta^-(v)} w_a = \sum_{a\in\delta^+(v)} w_a \qquad v \in V \cup \{v_a\} \tag{9.17}$$

$$y_a \leq w_a \leq |A|\, y_a \qquad a \in A \cup A_1 \tag{9.18}$$

$$y_a \leq w_{\bar{a}} \qquad a \in A\ (g(a) = (i,j)), \bar{a} = (i, v_a) \tag{9.19}$$

$$\sum_{a\in A_2} y_a = 1 \tag{9.20}$$

$$x_a \in \{0,1\} \qquad a \in A_R \tag{9.21}$$

$$y_a \geq 0 \text{ integer} \qquad a \in A' \tag{9.22}$$

$$w_a \geq 0 \qquad a \in A' \tag{9.23}$$

Die Lösung des Problems (9.13)-(9.23) als Pricing-Problem könnte eine Alternative zur Lösung der RCSPP im Pricing-Priblem sein. Aus Zeitmangel konnte diese Alternative noch nicht getestet werden.

9.4 Aufteilung in Distrikte

Eine weitere Möglichkeit das Problem zu lösen bestünde darin, das Straßennetzwerk in kleinere Distrikte aufzuteilen, wie es in der Praxis üblich ist. Es könnte dann das Straßenwinterdienstproblem für jedes Depot über einem wesentlich kleineren Graphen gelöst werden. In einem solchen Ansatz sollte in eine gute Aufteilung des Straßennetzes investiert werden. Wird zum Beispiel das Netzwerk in drei ungefähr gleich große Distrikte aufgeteilt, kann es sein, dass für jeden Distrikt zwei Fahrzeuge benötigt werden. Es wäre aber unter Umständen möglich mit nur vier Fahrzeugen auszukommen, wenn ein Distrikt größer als die beiden anderen ist.
Verfaren zur Aufteilung eines Straßennetzwerkes in Distrikte für den Straßenwinterdienst wurden von Kandula und Wright [16] und Muyldermans et al. [22] vorgestellt.

10 Zusammenfassung und Ausblick

Ziel dieser Arbeit war die Entwicklung eines Branch-and-Price-Verfahrens für den Straßenwinterdienst. In Kapitel 8 wurde das Verfahren erfolgreich für ein Beispielproblem angewendet.
Das Straßenwinterdienstproblem besteht darin Touren für Winterdienstfahrzeuge durch ein Straßennetz zu finden, so dass alle Richtungsspuren von genau einem Fahrzeug geräumt werden. Die Fahrzeuge sind dabei an verschiendenen Meistereien stationiert. Jede Fahrzeugtour muss an einem Depot beginnen und am selben Depot wieder enden. An den Straßenwinterdienst wird außerdem die Anforderung gestellt, dass alle Richtungsspuren innerhalb von drei Stunden geräumt werden. Geht man von einer Durchschnittsgeschwindigkeit von 30 km/h während eines Räumeinsatzes aus, kann keine Fahrzeugtour länger als 90 km sein.

Das Problem kann als CARP formuliert werden. Durch Dekomposition des Problems erhält man ein Set-Partitioning-Masterproblem sowie für jedes Fahrzeug ein RCSPP Subproblem. Sind alle Fahrzeuge identisch, können die Subprobleme zu einem einzigen Subproblem aggregiert werden. Die Spalten des Masterproblems korrespondieren dabei mit den zulässigen Fahrzeugtouren. Da es zu viele Fahrzeugtouren gibt, um diese alle explizit zu betrachten, wird das Masterproblem mittels Spaltenerzeugung gelöst. Zulässige Touren werden durch Lösung eines Ressource-Constrained-Shortest-Path-Problem erzeugt. Um das Verfahren zu starten wird eine Savingsheuristik verwendet, die eine zulässige Lösung für das Straßenwinterdienstproblem errechnet.

Für kleinere Probleme liefert das Branch-and-Price-Verfahren schnell die optimale Lösung. Für sehr große Straßennetzwerke ist die Rechenzeit allerdings zu lange und das Verfahren in dieser einfachen Form nicht durchführbar. Die Schwierigket liegt wahrscheinlich im Pricing-Problem. Das hier im Pricing verwendete RCSPP ist $\mathcal{NP}$-schwer. Um dennoch mittels Branch-and-Price eine Lösung zu finden, kann das Verfahren als Heuristik verwendet werden, indem nur bis zu einer bestimmten Knotentiefe neue Spalten erzeugt werden und dann versucht wird unter diesen Spalten eine möglichst gute zulässige Lösung zu finden.

In der Praxis ist es für den Straßenwinterdienst außerdem üblich das Straßennetzwerk erst in Distrike zu unterteilen und dann das Routing-Problem auf sehr viel kleineren Graphen zu lösen. Der Aufteilung des Straßennetzwerkes sollte bei einem solchen Ansatz einige Aufmerksamkeit gewidmet werden, da die Güte der Lösung nicht zuletzt von dieser Aufteilung abhängt.

10.1 Ausblick

In der Praxis gibt es sehr viele Probleme bei der Ausführung des Straßenwinterdienstes, die in dieser Arbeit noch nicht berücksichtigt wurden. Es wurde nur der Räumeinsatz betrachtet, in der Praxis ist es jedoch üblich, dass kombinierte Räum- und Streueinsätze gefahren

werden, d.h. ein Fahrzeug räumt und streut die Straße gleichzeitig. Wird auch der Streueinsatz in Betracht gezogen, muss das Modell erweitert werden. Die Streugutkapazitäten der Fahrzeuge müssen berücksichtigt werden. Da in der Regel die Fahrzeugflotte nicht homogen ist, kann nicht davon ausgegangen werden, dass die Fahrzeuge gleiche Kapazitäten besitzen.

Damit die Fahrzeuge längere Touren fahren können sind innerhalb des Straßennetzes neben den Meistereien weitere Streugutdepots errichtet worden, die in die Formulierung des Problems miteinbezogen werden sollten. Der Streueinsatz sollte außerdem für mehrspurige Straßen so geplant werden, dass alle Spuren einer Richtung simultan geräumt werden, damit keine Gefahren beim Spurwechsel auftreten.

An manchen Kreuzungen innerhalb des Straßennetzes ist es verboten, links abzubiegen oder umzukehren. Die Kreuzungen müssen daher so modelliert werden, dass Einschränkungen beim Abbiegen berücksichtigt werden können.

Literatur

[1] Leistungsheft für den Straßenbetriebsdienst auf Bundesfernstraßen Leistungsbereich 5: Winterdienst. `http://www.strassenwaerter.de/Leistungsheft_5_Winterdienst.pdf` , 2004.

[2] C. Barnhart, E.L. Johnson, G.L. Nemhauser, M.W.P. Savelsbergh, und P.H. Vance. Branch-and-Price: Column Generation for Solving Huge Integer Programs. *Operations Research*, 46:316 ff, 1998.

[3] E Benavent, V. Campos, A. Corberán, und E. Mota. The Capacitated Arc Routing Problem. A Heuristik Algorithm. *Questiió*, 14:107–122, 1990.

[4] BMVBW. Maßnahmen-Katalog zur Verbesserung der Wirtschaftlichkeit des Straßenbetriebsdienstes MK 6a. Allgemeines Rundschreiben Straßenbau Nr. 27/2004, 2004.

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

[6] G.B. Dantzig und P. Wolfe. Decomposition principle for linear programs. *Operations Research*, 8:101–111, 1960.

[7] M. Dür, A. Martin, und S. Ulbrich. Einführung in die Optimierung. Skript zur Vorlesung im WS 2008/2009.

[8] M. Dror. Note on the Complexity of the Shortest Path Models for Column Generation in VRPTW. *Operations Research*, 42:977–978, 1994.

[9] Y. Dumas, J. Desrosiers, und F. Soumis. The pickup and delivery problem with time windows. *European Journal of Operational Research*, 54:7–22, 1991.

[10] R.W. Eglese. Routeing winter gritting vehicles. *Discrete Applied Mathematics*, 48: 231–244, 1994.

[11] B.L. Golden und R.T. Wong. Capacitated arc routing problems. *Networks*, 11:305–315, 1981.

[12] Hessische Straßen- und Verkehrsverwaltung. Informationen über das Amt für Straßen- und Verkehrswesen Dillenburg. `http://www.hsvv.hessen.de/irj/HSVV_Internet?cid=4a8883b61008dd3f2ac331c4778e95ac` .

[13] S. Irnich und G. Desaulniers. Shortest Path Problems with Resource Constraints. In G. Desaulniers, J. Desrosiers, und M.M. Solomon, editors, *Column Generation*, pages 33–65. Springer, 2005.

[14] D.B. Johnson. Efficient Algorithms for Shortest Paths in Sparse Networks. *Journal of ACM*, 24:1–13, 1977.

[15] E. Johnson. Algorithms and model formulations in mathematical programming. chapter Modelling and strong linear programs for mixed integer programming, pages 1–43. Springer-Verlag New York, Inc., 1989.

[16] P Kandula und J.R. Wright. Designing network partitions to improve maintenance routing. *Journal of Infrastructure Systems*, 3:160–168, 1997.

[17] M.E. Lübbecke und J. Desrosiers. A Primer in Column Generation. In G. Desaulniers, J. Desrosiers, und M.M. Solomon, editors, *Column Generation*, pages 1–32. Springer, 2005.

[18] A.N. Letchford und A. Oukil. Exploiting sparsity in pricing routines for the capacitated arc routing problem. *Computers and Operations Research*, 36:2320 – 2327, 2009.

[19] H. Longo, M.P. de Aragão, und E. Uchoa. Solving capacitated arc routing problems using a transformation to the cvrp. *Computers and Operations Research*, 33.

[20] Marco E. Lübbecke und Jacques Desrosiers. Selected Topics in Column Generation. *Operations Research*, 53:1007–1023, 2005.

[21] A. Martin. Diskrete Optimierung. Skript zur Vorlesung im SS 2006.

[22] L. Muyldermans, D. Cattrysse, D. Van Oudheusdena, und T. Lotan. Districting for salt spreading operations. *European Journal of Operational Research*, 139:521–532, 2002.

[23] G.L. Nemhauser und L.A. Wolsey. *Integer and Combinatorial Optimization*. Wiley Intersience, 1999.

[24] N. Perrier, A. Langevin, und J.F. Campbell. A survey of models and algorithms for winter road maintenance: Part iv: Vehicle routing and fleet sizing for plowing and snow disposal. *Computers and Operations Research*, 34:258–294, 2007.

[25] N Perrier, A. Langevin, und A. Amaya. Vehicle Routing for Snow Plowing Operations. *Transportation Science*, 42:44–56, 2008.

[26] N. Perrier, A. Langevin, und Amayo C.A. Vehicle Routing for Urban Snow Plowing Operations. *Transportation Science*, 42:44–56, 2008.

[27] D.M. Ryan und B.A. Foster. An integer programming approach to scheduling. In *Computer Scheduling of Public Transport*, pages 269–280. North Holland Publishing Company, 1981.

[28] A. Schrijver. *Theory of Linear and Integer Programming*. Wiley Interscience, 1999.

[29] F. Vanderbeck. On Dantzig-Wolfe Decomposition in Integer Programming and ways to Perform Branching in a Branch-and-Price Algorithm. *Operations Research*, 48: 111–128, 2000.

[30] L.A. Wolsey. *Integer Programming.* Wiley-Interscience, 1998.

Erklärung

Hiermit versichere ich, dass ich die vorliegende Arbeit selbstständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt habe, dass alle Stellen der Arbeit, die wörtlich oder sinngemäß aus anderen Quellen übernommen wurden, als solche kenntlich gemacht sind und dass die Arbeit in gleicher oder ähnlicher Form noch keiner Prüfungsbehörde vorgelegt wurde.

Darmstadt, den 1. Dezember 2010