Train Timetabling Problem

Fahrplanoptimierung
Bachelor-Thesis von Dominique Achard aus Friedberg
September 2010

TECHNISCHE
UNIVERSITÄT
DARMSTADT

Fachbereich Mathematik
AG Optimierung

Train Timetabling Problem
Fahrplanoptimierung

Vorgelegte Bachelor-Thesis von Dominique Achard aus Friedberg

1. Gutachten: Dr. habil. Marko Lübbecke
2. Gutachten: Prof. Dr. Stefan Ulbrich

Tag der Einreichung:

Erklärung zur Bachelor-Thesis

Hiermit versichere ich, die vorliegende Bachelor-Thesis ohne Hilfe Dritter nur mit den angegebenen Quellen und Hilfsmitteln angefertigt zu haben. Alle Stellen, die aus Quellen entnommen wurden, sind als solche kenntlich gemacht. Diese Arbeit hat in gleicher oder ähnlicher Form noch keiner Prüfungsbehörde vorgelegen.

Darmstadt, den 22. September 2010

(Dominique Achard)

Inhaltsverzeichnis

1 Einleitung

Als im Jahr 1825 die öffentliche „Eisenbahn“ auf der Strecke zwischen Stockton und Darlington in England ihren Dienst begann, ahnte noch keiner, wozu dies einmal führen würde und welch wichtige Rolle die Eisenbahn einmal spielen wird. Heutzutage werden jährlich allein im deutschen Nahverkehr ca. 2.2 Milliarden Personen befördert (Quelle: stat. Bundesamt). Dies führt jedoch auch zu Problemen. Wie lassen sich die vielen tausend Züge aufeinander abstimmen, so dass es zu keinen Unfällen kommt? Wie lässt sich der Nutzen für die Kunden optimal damit verbinden? Wie lassen sich Anschlüsse aufrechterhalten? Wie lässt sich zudem die Haltezeit in Bahnhöfen minimieren?
Einem solchen Problem wollen wir uns in dieser Arbeit etwas nähern. Das geschieht auf Grundlage der Publikation „Modeling and solving the train timetabling problem“ von Alberto Caprara, Matteo Fischetti und Paolo Toth [1], die sich mit dem Problem eines konsistenten Fahrplans der Züge beschäftigt haben. Jedoch beschränken sie sich dabei auf die Betrachtung einer Strecke zwischen zwei „Metropolen“, auf welcher es jedoch Zwischenhalte geben kann. Da in den meisten Fällen Hin- und Rückrichtung unabhängig voneinander sind, kann man sich wiederum auf eine gerichtete Strecke beschränken. Desweiteren muss bei der Problemlösung beachtet werden, dass die Streckenkapazität eingehalten wird, sowie dass einige zusätzliche Bedingungen erfüllt sind. Dies wäre zum Beispiel, dass das Überholen zwischen den Bahnhöfen nicht möglich ist, sondern nur innerhalb eines Bahnhofs. Auch muss man beachten, dass es zum Beispiel eine gewisse Zeit dauert bis die Weichen umgestellt sind bzw. die Einfahrt in einen Bahnhof wieder zugelassen ist (selbes gilt auch beim Verlassen des Bahnhofs). Dies führt dazu, dass man gewisse Pufferzeiten einführen muss, die gewährleisten, dass diese Tätigkeiten auch korrekt durchgeführt werden können. All diese Bedingungen wollen wir mit einem mathematischen Modell beschreiben.
Dieses Modell basiert auf der Graphentheorie. Dabei werden Abfahrt bzw. Einfahrt zu einer gewissen Zeit mit einem Knoten verbunden. Daraus ergibt sich schließlich ein ganzzahlig lineares Programm. Darauf aufbauend lässt sich anschließend mit Hilfe von Lagrangerelaxation das Problem auf ein Multi-Commodity-Flow-Problem zurückführen, womit ein heuristisches Verfahren entwickelt werden kann, welches uns eine Lösung liefert (also einen Zeitplan für eine gerichtete Strecke). Wir werden zunächst einmal damit beginnen, das Problem passend zu beschreiben und anschließend zeigen, dass es NP-schwer ist, um zu gewährleisten, dass es sich nicht exakt mit einem polynomialen Algorithmus lösen lässt und somit unseren heuristischen Ansatz begründet.

2 Eine erste mathematische Beschreibung

In diesem Kapitel versuchen wir nun das Problem aus der Realität in eine möglichst gute mathematische Form zu bringen, um einen Lösungsalgorithmus zu entwickeln.

2.1 Train Timetabling Problem

Da wir unser Problem in einem gewissen Planungszeitraum betrachten, macht es Sinn diesen zunächst einmal geschickt zu definieren. Der Planungszeitraum ist in **Perioden** unterteilt, in denen sich der Fahrplan wiederholen soll (z.B. jeden Tag). Eine Periode wollen wir wie folgt definieren:

$$P := \{1, \dots, q\} \quad q \in \mathbb{N}$$

Wie man sieht besteht eine Periode aus mehreren **Zeitpunkten**, die sich wie gewünscht wählen lassen (z.B. Minuten).

Anschließend wollen wir unsere Strecke charakterisieren. Dazu wird die Strecke in ihre Zwischenstationen unterteilt und mit einer sortierten Menge der **Bahnhöfe**, wie folgt veranschaulicht:

$$S := \{1, \dots, s\} \quad s \in \mathbb{N}$$

Da auf der Strecke eine gewisse Anzahl an Zügen fahren soll, benötigen wir hierfür eine brauchbare mathematische Beschreibung. Die möglichen **Züge**, die diese Strecke in einer Periode benutzen können, beschreiben wir durch eine endliche Menge:

$$T := \{1, \dots, t\} \quad t \in \mathbb{N}$$

Ein Zug, der auf dieser Strecke fährt, muss jedoch nicht unbedingt in dem ersten Bahnhof seine Fahrt beginnen. Dies motiviert die folgende Notation.

Notation 2.1. *Sei $j \in T$. Wir nennen f_j den Bahnhof, in dem der Zug j seine Fahrt auf der Strecke S beginnt* ***(Eintrittsbahnhof)****. Desweiteren nennen wir l_j den Bahnhof, in dem die Fahrt des Zuges j auf der Strecke S endet* ***(Austrittsbahnhof)****.*

Wir definieren eine sortierte Menge der **anzufahrenden Bahnhöfe** für den jeweiligen Zug

$$S^j := \{f_j, \dots, l_j\} \subseteq S \quad j \in T$$

Bemerkung. *Solche Fälle treten auf, wenn es „Quereinsteiger“ gibt, welche die Strecke ab dem z.B. dritten Bahnhof und bis zum fünften Bahnhof benutzen und sich anschließend wieder anders orientieren (eine andere Strecke benutzen). Des Weiteren kann ein Zug auch Bahnhöfe auslassen.*

Wir wenden uns nun dem Fahrplan zu, der den Output unseres Problems darstellt.

Definition 2.1. *Ein* ***Fahrplan*** *weist jedem Zug $j \in T$ eine Abfahrtszeit in f_j sowie eine Ankunftszeit in l_j zu. Des Weiteren werden auch in allen Zwischenstationen $t_0 \in S^j \backslash \{f_j, l_j\}$ Ankunfts- sowie Abfahrtszeiten zugewiesen.*
Wir nennen die ***Menge aller Fahrpläne*** *$\mathscr{F}$.*

Bemerkung. *Der Fahrplan ist periodisch für jeden Zug, d.h. er wiederholt sich nach einer Periode innerhalb des Planungszeitraums immer wieder. Aus diesem Grund entwickeln wir nur eine Lösung für eine Periode.*

Definition 2.2. *Die* ***Fahrtzeit*** *ist die Zeit zwischen der Abfahrt in Bahnhof f_j und der Ankunft in Bahnhof l_j.*

Wir weisen jedem Zug in einem Fahrplan einen abstrakten **Nutzen** zu

$$N : \mathscr{F} \times T \to \mathbb{R}$$

Daraus abgeleitet wird der Nutzen N_f für einen Fahrplan $f \in \mathscr{F}$

$$N_f := \sum_{j \in T} N(f, j)$$

Mit Hilfe der zuvor genannten Definitionen können wir unser betrachtetes Problem nun angeben.

Train Timetabling Problem (TPP) 2.1.
Input: *Strecke S, Züge T, für alle Züge $j \in T$ die anzufahrenden Bahnhöfe S^j, Periode P, mögliche Beschränkungen*
Output: *Menge der nutzenoptimalen Fahrpläne*

Notation 2.2. *Wir benötigen nun für jeden Zug $j \in T$ zugspezifische Parameter.*

(i) *den **idealen Nutzen** Π_j, falls der Zug j ohne Einschränkungen fahren kann*
(ii) *die **ideale Abfahrtszeit** $\overline{D}_j$ des Zuges j in f_j*
(iii) *die **ideale Haltezeit** $\overline{S}_{ij}$ des Zuges j im Bahnhof $i \in S^j \backslash \{f_j, l_j\}$*
(iv) *die **ideale Fahrtzeit** $\overline{F}_{ij}$ des Zuges j auf der Strecke zwischen Bahnhof i und i+1 mit $i \in S^j \backslash \{l_j\}$*
(v) *die **ideale Fahrtzeit** $\mathscr{L}_j$ des Zuges j, wobei*

$$\mathscr{L}_j := \sum_{i \in S^j \backslash \{f_j, l_j\}} \overline{S}_{ij} + \sum_{i \in S^j \backslash \{l_j\}} \overline{F}_{ij}$$

Bemerkung. $\sum_{j \in T} \Pi_j$ *bildet eine obere Schranke für den Gesamtnutzen eines Fahrplans. Das Ziel ist nun möglichst Nahe an diese Schranke heranzukommen.*

2.2 Streckenbeschränkungen

Wie bereits in der Einleitung erwähnt wurde, ist es unmöglich, dass mehrere Züge gleichzeitig auf einer Strecke fahren. Mit solchen Streckenbeschränkungen wollen wir uns in diesem Teil beschäftigen und geben einen Überblick, den wir in den späteren Kapiteln präzisieren werden. Dazu müssen wir uns jedoch zunächst klar machen, was für Beschränkungen überhaupt beachtet werden müssen. Also fangen wir auch hier wieder bei dem realen Problem an und stellen uns die Frage: Was ist erlaubt und was nicht? Dabei unterscheiden wir zwischen zwei unterschiedlichen Typen von Beschränkungen und beginnen mit den Streckenkapazitätsbeschränkungen.

Notation 2.3. *Sei $i \in S$.*

(i) *Eine **untere Schranke** $a_i \in \mathbb{N}$ für die **Ankunft** ist der Abstand, der mindestens zwischen zwei in den Bahnhof i einfahrenden Zügen eingehalten werden muss.*
(ii) *Eine **untere Schranke** $d_i \in \mathbb{N}$ für die **Abfahrt** ist der Abstand, der mindestens zwischen zwei den Bahnhof i verlassenden Zügen eingehalten werden muss.*

Definition 2.3. *Eine **Streckenkapazitätsbeschränkung** ist eine Nebenbedingung, die folgendes gewährleistet*

(i) *Die unteren Schranken für die Ankunft werden eingehalten.*
(ii) *Die unteren Schranken für die Abfahrt werden eingehalten.*
(iii) *Überholen ist nur in Bahnhöfen erlaubt.*

Bemerkung. *Züge dürfen auch in nicht geplanten Haltebahnhöfen halten.*

Wir wenden uns nun den Zeitfensterbeschränkungen zu. Diese beschäftigen sich damit, dass gewisse Anschlussbedingungen erfüllt sein müssen, damit zum Beispiel Anschlusszüge für die Kunden gewährleistet werden.

Notation 2.4. *Sei $i \in T$ und $j \in S$.*

(i) *$arr_{ij} \in P$ ist der **spätestmögliche Ankunftszeitpunkt** des Zuges i im Bahnhof j.*
(ii) *$dep_{ij} \in P$ ist der **frühstmögliche Abfahrtszeitpunkt** des Zuges i im Bahnhof j.*

Definition 2.4. *Eine **Zeitfensterbeschränkung** ist eine Nebenbedingung, die folgendes gewährleistet:*

(i) *Der spätmöglichste Ankunftszeitpunkt wird eingehalten.*
(ii) *Der frühstmögliche Abfahrtszeitpunkt wird eingehalten.*

Hiermit schließen wir die Betrachtung der Streckenbeschränkungen zunächst ab. Sie werden später noch eine entscheidende Rolle spielen.

2.3 Nutzen

Die Fragestellung mit der man diesen Abschnitt überschreiben könnte, lautet wie folgt: Was wird versucht zu optimieren? Dazu gehen wir auf den im vorherigen Kapitel definierten Nutzen näher ein. Hierbei benötigen wir zunächst ein paar Definitionen.

Definition 2.5. *Sei $j \in T$. Der **Shift** $V_j : \mathscr{F} \to \mathbb{R}$ beschreibt die Differenz der idealen Abfahrtszeit $\overline{F}_j$ zu der Abfahrtszeit in f_j gemäß des Fahrplans $f \in \mathscr{F}$ des Zuges j.*

Definition 2.6. *Sei $j \in T$. Der **Stretch** $\mu_j : \mathscr{F} \to \mathbb{R}$ beschreibt die Differenz der idealen Fahrtzeit $\mathscr{L}_j$ zu der Fahrtzeit im aktuellen Fahrplan $f \in \mathscr{F}$ des Zuges j.*

Wir modellieren unseren Nutzen eines Zuges $j \in T$ in einem Fahrplan $f \in \mathscr{F}$ so, dass er von den folgenden drei Parametern abhängt:

(i) dem idealen Nutzen Π_j
(ii) dem Shift $V_j(f)$
(iii) dem Stretch $\mu_j(f)$

Die Idee ist nun, von dem idealen Nutzen Bestrafungen für Stretch und Shift abzuziehen und somit einen Nutzen zu erhalten.

Dies motiviert abhängig von dem Shift eine Funktion zu definieren, die die verspätete Abfahrt in eine Nutzenbestrafung transformiert.

Definition 2.7. *Sei $j \in T$. Die Funktion $\phi_j : P \cup \{0\} \to \mathbb{R}$ ist eine vom Benutzer definierte, nicht-fallende **Bestrafungsfunktion für den Shift** mit $\phi_j(0) = 0$. Diese Funktion ist für jeden Zug j spezifisch.*

Wir gehen bei dem Stretch nun davon aus, dass dieser eine lineare Bestrafung besitzt und wir somit nur einen Parameter γ_j zur Beschreibung benötigen (auch hier sollte keine Bestrafung bei keinem Stretch vorliegen).

Definition 2.8. *Sei $j \in T$. Der Parameter $\gamma_j \geq 0$ gibt die **Bestrafung des Stretchs** an. Dieser ist für jeden Zug j spezifisch.*

Wie oben erwähnt definieren wir unsere Nutzenfunktion also ausgehend vom idealen Nutzen und ziehen die entsprechenden Bestrafungen ab.

$$N(f, j) := \Pi_j - \phi_j(V_j(f)) - \gamma_j \cdot \mu_j(f)$$

Bemerkung.

(i) *Die Nutzenfunktionen sind normalerweise bei dem selben Zugtyp zur selben Zeit identisch.*
(ii) *Bei negativem Nutzen könnte man sich überlegen, den Zug überhaupt nicht fahren zu lassen, da die Bestrafung höher ist als der Nutzen. Dies stellt eine implizite Zeitfensterbeschränkung dar.*

Damit schließen wir die mathematische Modellierung zunächst ab und gehen im nächsten Kapitel darauf ein, ob sich das Problem „einfach/effizient“ lösen lässt.

3 Komplexität des Problems

In diesem Kapitel wollen wir nun zeigen, dass das Problem NP-schwer [2] ist. Wenn uns dies gelungen ist, können wir damit das Anwenden einer Heuristik begründen, da es keinen „einfachen“ polynomialen Algorithmus geben wird.
Unsere Idee ist es eine Abbildung zwischen unserem Problem und dem MISP (Max-Independent Set Problem [3]) zu finden, um somit nachzuweisen, dass das Problem NP-schwer ist. Zu diesem Zweck führe ich das MISP zunächst ein.

3.1 Das Max-Independent Set Problem (MISP)

Das Max-Independent Set Problem sucht in einem ungerichteten Graphen H=(N,E) nach einer Knotenmenge mit maximaler Kardinalität, so dass zwischen zwei beliebigen Knoten in dieser Menge keine Kante in G existiert.

Problem 3.1. ***Max-Independent Set Problem***

Input: *ein ungerichteter Graph H = (N,E)*

Output: *eine Knotenmenge $S \subseteq N$ mit maximaler Kardinalität $|S|$, für die gilt*

$$(i,j) \notin E \; \forall i,j \in S$$

Dieses Problem ist NP-schwer [3], was uns dazu motiviert eine Reduktion von diesem Problem auf unser TTP zu entwickeln. Damit hätten wir gezeigt, dass das TTP NP-schwer ist.

3.2 Beweis NP-schwer

Dieses Kapitel besteht im Kern aus der polynomiellen Reduktion und somit aus dem Beweis des folgenden Satzes.

Satz 3.1. *Jede MISP-Instanz lässt sich polynomiell in eine TTP-Instanz transformieren, wobei die korrespondierenden Lösungen den gleichen Wert haben.*

Beweis.

1) **Notation.**
Sei $H = (N, E)$ der Graph der MISP-Instanz mit $E = \{e_1, \ldots, e_m\}$ und $e_i = (j_i, k_i)$ für alle $i = 1, \ldots, m$ mit $m \in \mathbb{N}$. Wir weisen jedem Knoten in N einen Zug zu, d.h. $t = |N|$ und setzen

$$\Pi_j = 1; \quad \phi_j(0) = 0; \quad \phi_j(x) = 1 \; \forall x \neq 0; \quad \gamma_j = 1$$

Aus diesen Notationen folgt schon, dass ein Zug sofort ausfällt, wenn er nicht ideal geplant werden kann, also ein Shift oder Stretch auftritt.

$$N(f,j) = 1 - V_j(f) - \mu_j(f)$$

Da wir Minuten betrachten und somit $V_j(f), \mu_j(f) \in \mathbb{N}$ gilt, muss $V_j(f) = \mu_j(f) = 0$ gelten, damit der Zug geplant wird. Da alle Züge, wenn sie geplant werden, auch den gleichen Nutzen von 1 haben, sucht das entwickelte TTP nach einer maximalen Anzahl von Zügen, die kompatible ideale Fahrpläne haben.
Wir setzen nun genau $m+1$ Bahnhöfe mit $f_j = 1$ und $l_j = m+1$ für alle $j \in T$. Zusätzlich setzen wir $a_i = d_i = 1$ für alle $i \in S$. Die Fahrtzeit zwischen zwei Bahnhöfen setzen wir auf 1.

2) **Entwicklung der idealen Fahrpläne.**
Wir nennen nun zwei Züge inkompatibel, wenn ihre idealen Fahrpläne nicht zusammen geplant werden können, woraus folgt, dass nur einer der beiden Züge geplant werden kann.
Dies berücksichtigen wir, indem wir zwei Züge j,k als inkompatibel kennzeichnen, wenn eine Kante $(j,k) \in E$ existiert. Wenn wir also eine Kante $e_i = (j_i, k_i)$ haben, heißt das, dass im idealen Fahrplan die Züge j_i und k_i zur selben Zeit im Bahnhof i abfahren und somit auch zur selben Zeit im Bahnhof i+1 ankommen. Die i-te Kante charakterisiert somit die Inkompatibilität auf der Strecke zwischen i und i+1.
Somit lassen sich nun aus dem Graphen die idealen Fahrpläne aller Züge entwickeln. Wir beginnen mit der ersten Kante $e_1 = (j_1, k_1)$. Wir setzen die Abfahrtszeit der Züge j_1 und k_1 in dem Bahnhof 1 auf den Wert j_1. Für die weiteren Züge ($j \neq j_1$ und $j \neq k_1$) setzen wir die Abfahrtszeit auf den Wert j. Die Ankunftszeit in Bahnhof 2 ist jeweils eine Minute nach der Abfahrt in 1.
Wir gehen genauso für eine allgemeine Kante $e_i = (j_i, k_i)$ vor. Dabei setzen wir die Abfahrtszeit der Züge j_i und k_i im Bahnhof i auf

$$(i-1) \cdot (t+1) + j_i$$

Für alle Züge $j \neq j_i$ und $j \neq k_i$ wird die Abfahrtszeit in i wie folgt gesetzt

$$(i-1) \cdot (t+1) + j$$

Die Ankunftszeit im Bahnhof i+1 ist wiederum eine Minute nach der Abfahrt in i. Der Term $(i-1) \cdot (t+1)$ berücksichtigt den Bahnhof in dem wir uns befinden.
Somit lassen sich iterativ die idealen Fahrpläne jedes Zuges entwickeln.

3) **Lösung.**
Da auf Grund der Konstruktion der Zug nur ideal fahren kann oder ausfällt, suchen wir bei der entstandenen TTP-Instanz die maximale Anzahl an zueinander kompatiblen Fahrplänen. Dies entspricht genau dem Suchen der maximal-unabhängigen Knotenmenge in unserer MISP-Instanz, da inkompatible Züge (Knoten) genau mit einer Kante verbunden sind. Auch der Lösungswert stimmt überein, da der Nutzen eines geplanten Zuges genau 1 entspricht.
Uns ist also eine Reduktion von einer MISP-Instanz zu einer TTP-Instanz gelungen.

□

Die oben beschriebene Reduktion lässt sich jetzt an einem Beispiel verdeutlichen, welches helfen soll diese besser zu verstehen.

Beispiel. *Sei* $H = (N, E)$ *mit* $N = \{1, 2, 3, 4\}$ *und* $E = \{(1,2), (1,3), (1,4), (2,3), (2,4)\}$.
Somit werden die Abfahrtszeiten wie folgt ermittelt.

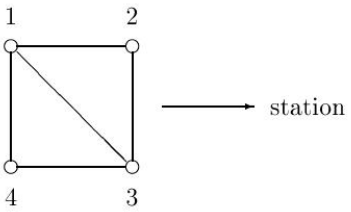
Abfahtszeiten in 1: mit $i = 1$ *gilt* $(i-1) \cdot (t+1) = 0 \cdot 5 = 0$
1 und 2: $0 + 1 = 1$
3: $0 + 3 = 3$
4: $0 + 4 = 4$

Abfahtszeiten in 2: mit $i = 2$ *gilt* $(i-1) \cdot (t+1) = 1 \cdot 5 = 5$
1 und 3: $5 + 1 = 6$
2: $5 + 2 = 7$
4: $5 + 4 = 9$

Abfahtszeiten in 3: mit $i = 3$ *gilt* $(i-1) \cdot (t+1) = 2 \cdot 5 = 10$
1 und 4: $10 + 1 = 11$
2: $10 + 2 = 12$
3: $10 + 3 = 13$

Abfahtszeiten in 4: mit $i = 4$ *gilt* $(i-1) \cdot (t+1) = 3 \cdot 5 = 15$
1: $15 + 1 = 16$
2 und 3: $15 + 2 = 17$
4: $15 + 4 = 19$

Abfahtszeiten in 5: mit $i = 5$ *gilt* $(i-1) \cdot (t+1) = 4 \cdot 5 = 20$
1: $20 + 1 = 21$
2: $20 + 2 = 22$
3 und 4: $20 + 3 = 23$



$H = (N, E)$

		train			
		1	2	3	4
1	dep.	1	1	3	4
2	arr.	2	2	4	5
	dep.	6	7	6	9
3	arr.	7	8	7	10
	dep.	11	12	13	11
4	arr.	12	13	14	12
	dep.	16	17	17	19
5	arr.	17	18	18	20
	dep.	21	22	23	23
6	arr.	22	23	24	24

Wir können nun in einem abschließenden Satz zeigen, dass das TTP nicht polynomiell lösbar ist.

Satz 3.2. *Es existiert kein Algorithmus A mit dem sich eine beliebige TTP-Instanz polynomiell lösen lässt.*

Beweis.
Wir werden hier einen Widerspruchsbeweis verwenden.
Annahme: Es existiert ein polynomieller Algorithmus A für das TTP.

Sei I eine beliebige MISP-Instanz. Aus dem Satz 3.1 folgt, dass sich die Instanz I polynomiell in eine TTP-Instanz J reduzieren lässt.
Nach der Annahme lässt sich J und somit auch I mit A lösen. Dies ist jedoch ein Widerspruch dazu, dass MISP in NP-schwer liegt. □

4 Graphenmodellierung

In diesem Kapitel definieren wir einen gerichteten azyklischen Multigraphen $G = (V, A)$, wobei V die Knotenmenge darstellt und A die Kantenmenge darstellt,welche wir wir im Folgenden definieren werden.
Die Idee ist für jeden möglichen Ankunfts- bzw. Abfahrtszeitpunkt in jedem Bahnhof einen Knoten einzuführen und diese, wenn es möglich ist, mit Kanten zu verbinden, um anschließend für jeden Zug einen nutzenmaximalen Pfad in diesem Graphen zu suchen.

4.1 Knotenmenge

Wir beginnen mit der Knotenmenge V und führen dazu zunächst eine künstliche Quelle und Senke ein, die später den Start bzw. das Ende jedes Pfades darstellen soll.

Notation 4.1. *Sei σ die künstliche **Quelle** und τ die künstliche **Senke**.*

Im nächsten Schritt werden wir nun die Menge der Ankunftsknoten definieren.

Notation 4.2. *Sei $i \in S\backslash\{1\}$. Dann ist U^i die Menge der möglichen **Ankunftszeitpunkte** in i.*

Damit haben wir für jeden möglichen Ankunftszeitpunkt in Bahnhof i einen Knoten in U^i. In dem ersten Bahnhof gibt es keinen Ankunftszeitpunkt, da hier nur abgefahren wird. Selbiges gilt für die Abfahrt im letzten Bahnhof. Genauso gehen wir nun für die Abfahrtszeitpunkte vor.

Notation 4.3. *Sei $i \in S\backslash\{s\}$. Dann ist W^i die Menge der möglichen **Abfahrtszeitpunkte** in i.*

Definition 4.1. *Sei $u \in \bigcup_{i=1}^{s-1} U^i$ und $w \in \bigcup_{i=2}^{s} W^i$. Dann nennen wir u einen **Ankunftsknoten** und w einen **Abfahrtsknoten**.*

Bemerkung.

(i) *Sei $i, j \in S\backslash\{1\}$ mit $i \neq j$. Dann gilt $U^i \cap U^j = \emptyset$.*
(ii) *Sei $i, j \in S\backslash\{s\}$ mit $i \neq j$. Dann gilt $W^i \cap W^j = \emptyset$.*
(iii) *Sei $i \in S\backslash\{1\}$ $j \in S\backslash\{s\}$. Dann gilt $U^i \cap W^j = \emptyset$.*

Beweis. Dies folgt direkt aus der Definition der Mengen. □

Also haben wir eine Partition unserer Knotenmenge V entwickelt und können diese nun angeben.

$$V := \{\sigma, \tau\} \cup \bigcup_{i=1}^{s-1} U^i \cup \bigcup_{j=2}^{s} W^j$$

4.2 Knotenrelation

Da jeder Knoten mit einer Zeit in Verbindung gebracht wird, definieren wir zu den Knoten eine Abstandsfunktion. Damit können wir anschließend eine Relation auf der Knotenmenge angeben.

Definition 4.2. *Sei $v \in V$. Dann ist $\theta : V \to P$ die Zeitfunktion, die zu jedem Knoten die jeweilige **Zeitinstanz** liefert.*

Daraus entwickeln wir nun die Abstandsfunktion.

Definition 4.3. *Sei $u, v \in V\backslash\{\tau, \sigma\}$. Wir definieren die **Abstandsfunktion** $\Delta : V\backslash\{\tau, \sigma\} \times V\backslash\{\tau, \sigma\} \to P \cup \{0\}$ wie folgt*

$$\Delta(u, v) := \begin{cases} \theta(v) - \theta(u) & falls \quad \theta(v) \geq \theta(u) \\ \theta(v) - \theta(u) + q & sonst \end{cases}$$

Der zweite Fall berücksichtigt die Periodizität.

Bemerkung. *Sei $u, v, z \in V\backslash\{\tau, \sigma\}$.*

(i) *$\Delta(\cdot, \cdot)$ ist wohldefiniert. Es gilt sogar: $\Delta(V, V) \subseteq \{0, \ldots, q-1\}$.*
(ii) *$\Delta(\cdot, \cdot)$ stellt keine Metrik dar.*

(iii) *Es gilt eine Richtung der Definitheit:* $\Delta(u,u)=0$.

(iv) *Wie man im Beweis sieht, gilt die Symmetrie nicht, jedoch lässt sich eine Beziehung zwischen den beiden Abständen angeben:*

$$\Delta(u,v)=\begin{cases} q-\Delta(v,u) & ,\theta(u)\neq\theta(v) \\ \Delta(v,u) & ,\theta(u)=\theta(v) \end{cases}$$

(v) *Es gilt die Dreiecksungleichung:* $\Delta(u,v)\leq\Delta(u,z)+\Delta(z,v)$.

Beweis. Sei $u,v,z\in V$.

(i) Wir wollen nun zeigen, dass in jedem Fall gilt $\Delta(u,v)\in P\cup\{0\}$. Also $0\leq\Delta(u,v)\leq q$.
Wir wissen bereits $1\leq\theta(u)\leq q$ und $1\leq\theta(v)\leq q$, da $\theta(u),\theta(v)\in\theta(V)\subseteq P$.

1.Fall: $\theta(v)\geq\theta(u)$
$\Delta(u,v)=\theta(v)-\theta(u)\leq q-1<q$, da $\theta(v)\leq q$ und $\theta(u)\geq 1$.
$\Delta(u,v)=\theta(v)-\theta(u)\geq 0$, da $\theta(v)\geq\theta(u)$.

2.Fall: $\theta(u)>\theta(v)$
$\Delta(u,v)=\theta(v)-\theta(u)+q\leq q-1$, da $\theta(v)-\theta(u)<0$ und somit $\theta(v)-\theta(u)\leq -1$ da $\theta(u)\in\mathbb{N}$.
$\Delta(u,v)=\theta(v)-\theta(u)+q>0-q+q=0$, da $\theta(v)>0$ und $\theta(u)\leq q$

(ii) Dies scheitert bereits an der Symmetrie, da für OBdA $\theta(v)\geq\theta(u)$ gilt:

$$\Delta(u,v)=\theta(v)-\theta(u)\neq q-[\theta(v)-\theta(u)]=\theta(u)-\theta(v)+q=\Delta(v,u)$$

(iii) Die Aussage folgt direkt aus der Definition:

$$\Delta(u,u)=\theta(u)-\theta(u)=0$$

(iv) Wir unterscheiden hier die drei auftretenden Fälle:

1.Fall: $\theta(v)>\theta(u)$
$\Delta(u,v)+\Delta(v,u)=\theta(v)-\theta(u)+\theta(u)-\theta(v)+q=q$

2.Fall: $\theta(v)<\theta(u)$
$\Delta(u,v)+\Delta(v,u)=\theta(v)-\theta(u)+q+\theta(u)-\theta(v)=q$

3.Fall: $\theta(v)=\theta(u)$
$\Delta(u,v)=\theta(v)-\theta(u)=0$ und $\Delta(v,u)=\theta(u)-\theta(v)=0$
$\Rightarrow\Delta(u,v)=\Delta(v,u)$

Damit ist die Aussage bewiesen.

(v) Auch hier rechnen wir die sechs auftretenden Fälle durch.

1.Fall: $\theta(u)\leq\theta(z)\leq\theta(v)$
$\Delta(u,z)+\Delta(z,v)=\theta(z)-\theta(u)+\theta(v)-\theta(z)=\theta(v)-\theta(u)=\Delta(u,v)$

2.Fall: $\theta(u)\leq\theta(v)\leq\theta(z)$
$\Delta(u,z)+\Delta(z,v)=\theta(z)-\theta(u)+\theta(v)-\theta(z)+q=\theta(v)-\theta(u)+q=\Delta(u,v)+q\geq\Delta(u,v)$

3.Fall: $\theta(z)\leq\theta(v)\leq\theta(u)$
$\Delta(u,z)+\Delta(z,v)=\theta(z)-\theta(u)+q+\theta(v)-\theta(z)=\theta(v)-\theta(u)+q=\Delta(u,v)$

4.Fall: $\theta(z)\leq\theta(u)\leq\theta(v)$
$\Delta(u,z)+\Delta(z,v)=\theta(z)-\theta(u)+q+\theta(v)-\theta(z)=\theta(v)-\theta(u)+q=\Delta(u,v)+q\geq\Delta(u,v)$

5.Fall: $\theta(v)\leq\theta(z)\leq\theta(u)$
$\Delta(u,z)+\Delta(z,v)=\theta(z)-\theta(u)+q+\theta(v)-\theta(z)+q=\theta(v)-\theta(u)+2q=\Delta(u,v)+q\geq\Delta(u,v)$

6.Fall: $\theta(v)\leq\theta(u)\leq\theta(z)$
$\Delta(u,z)+\Delta(z,v)=\theta(z)-\theta(u)+\theta(v)-\theta(z)+q=\theta(v)-\theta(u)+q=\Delta(u,v)$

Damit haben wir die Dreiecksungleichung im allgemeinen Fall bewiesen.

□

Nun können wir nach den Vorbereitungen eine Ordnungsrelation auf der Menge der Knoten angeben. Die Intuition dabei ist, dass man bei der Reihenfolgenbetrachtung immer den kürzeren Abstand in Betracht zieht (Berücksichtigung der Periodizität).

Definition 4.4. *Sei $u, v \in V$. Wir definieren $\preceq$ wie folgt*

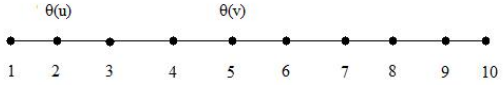
$$u \preceq v :\Longleftrightarrow \Delta(v,u) \geq \Delta(u,v)$$

Man spricht auch davon, dass der Knoten u dem Knoten v ***vorausgeht****.*

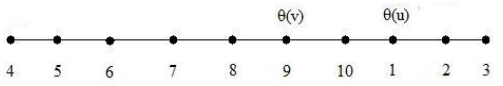
Wir wollen die Definition nun an zwei kleinen Beispielen veranschaulichen.

Beispiele. *Sei $u, v \in V$.*

(i) *Sei $\theta(u) = 2$ und $\theta(v) = 5$. Sei $q = 10$.*
Wir berechnen die Abstände zwischen den beiden Knoten.
$\Delta(u,v) = \theta(v) - \theta(u) = 5 - 2 = 3$ *und*
$\Delta(v,u) = 10 - \Delta(u,v) = 7$
Damit gilt $\Delta(u,v) \leq \Delta(v,u)$ und somit $u \preceq v$.



(ii) *Sei $\theta(u) = 1$ und $\theta(v) = 9$. Sei $q = 10$.*
Wir berechnen nun wieder die Abstände zwischen den beiden Knoten.
$\Delta(u,v) = \theta(v) - \theta(u) = 9 - 1 = 8$ *und*
$\Delta(v,u) = 10 - \Delta(u,v) = 2$
Damit gilt $\Delta(u,v) \geq \Delta(v,u)$ und somit $v \preceq u$.



Bemerkung. *Die Relation „$\preceq$“ ist reflexiv aber nicht transitiv.*

Beweis.
Reflexivität:
Dies folgt direkt aus $\Delta(u,u) = 0 \geq 0 = \Delta(u,u)$

Transitivität:
Hier entwickeln wir ein Gegenbeispiel.
Sei $q = 10$. Sei $u, v, w \in V$ mit $\theta(u) = 2, \theta(v) = 6$ und $\theta(w) = 10$.
Dann gilt:

- $\Delta(u,v) = 4 \leq 6 = \Delta(v,u) \Rightarrow u \preceq v$
- $\Delta(v,w) = 4 \leq 6 = \Delta(w,v) \Rightarrow v \preceq w$

Jedoch gilt auch:
$\Delta(u,w) = 8 \geq 2 = \Delta(w,u)$ und somit $w \preceq u$.
Dies ist ein Widerspruch zur Transitivität. □

4.3 Kantenmenge

Jetzt lässt sich auch eine Partition für die Kantenmenge A entwickeln. Das Ziel ist für jeden Zug $j \in T$ eine Kantenmenge A^j zu konstruieren. Aus unserer Definition des Graphen ergeben sich nun vier verschiedene Streckentypen, die wir zunächst definieren.

Definition 4.5. *Sei $j \in T$ und $v \in W^{f_j}$, wobei v zu einer möglichen Abfahrt des Zuges j im Bahnhof f_j gehört. Dann heißt (σ, v)* ***Startkante*** *in Bezug auf j.*
Des Weiteren nennen wir A_S^j die ***Menge aller Startkanten*** *in Bezug auf j.*

Definition 4.6. *Sei $j \in T$ und $i \in S^j \backslash \{f_j, l_j\}$. Sei $u \in U^i$ und $v \in W^i$, so dass*

(i) *die Zeitfensterbeschränkungen eine Ankunft von Zug j im Bahnhof i zur Zeit $\theta(u)$ zulassen.*

(ii) *die Zeitfensterbeschränkungen eine Abfahrt von Zug j im Bahnhof i zur Zeit $\theta(v)$ zulassen.*

(iii) $\Delta(u, v)$ *mindestens so groß ist, wie die minimale Haltezeit* $\overline{S}_{ij}$ *im Bahnhof i.*

Dann heißt (u, v) **Bahnhofskante** *in Bezug auf j im Bahnhof i.*
Des Weiteren nennen wir $A^j_{B,i}$ *die* **Menge aller Bahnhofskanten** *in Bezug auf j im Bahnhof i.*

Definition 4.7. *Sei* $j \in T$ *und* $i \in S^j \backslash \{l_j\}$. *Sei* $v \in W^i$ *und* $u \in U^{i+1}$, *so dass*

(i) *die Zeitfensterbeschränkungen eine Abfahrt von Zug j im Bahnhof i zur Zeit* $\theta(v)$ *erlauben*

(ii) *die Zeitfensterbeschränkungen eine Ankunft von Zug j im Bahnhof i zur Zeit* $\theta(u)$ *erlauben*

(iii) $\Delta(v, u)$ *mindestens so groß ist wie die minimale Fahrtzeit* $\overline{F}_{ij}$ *auf der Strecke zwischen i und i+1*

Dann heißt (v, u) **Streckenkante** *in Bezug auf j auf der Strecke zwischen i und i+1.*
Des Weiteren nennen wir $A^j_{Str,i}$ *die* **Menge aller Streckenkanten** *in Bezug auf j auf der Strecke zwischen i und i+1.*

Definition 4.8. *Sei* $j \in T$. *Sei* $u \in U^{l_j}$, *wobei u zu einer möglichen Ankunft des Zuges j im Bahnhof* l_j *gehören muss.*
Dann heißt (u, τ) **Endkante** *in Bezug auf j.*
Desweiteren nennen wir A^j_E *die* **Menge aller Endkanten** *in Bezug auf j.*

Jetzt lässt sich mit Hilfe dieser vier Kantentypen für jeden Zug eine eigene Kantenmenge definieren, die genau die Kanten enthält, die den entsprechenden Zug betreffen.

$$A^j := A^j_S \cup \left(\bigcup_{i \in S^j \backslash \{f_j, l_j\}} A^j_B, i \right) \cup \left(\bigcup_{i \in S^j \backslash \{l_j\}} A^j_{Str,i} \right) \cup A^j_E \quad j \in T$$

Nun lässt sich aus diesen Mengen wiederum die **Kantenmenge** unseres Graphen definieren.

$$A := \bigcup_{j=1}^{t} A^j$$

Bemerkung.

(i) *Der somit entstandene Graph* $G = (V, A)$ *ist nach Konstruktion azyklisch.*

(ii) *Alle Beschränkungen (Zeitfensterbeschränkungen,...), die sich auf den speziellen Zug beziehen, sind nun in der Struktur des Graphen G integriert. Somit müssen wir uns um diese im Weiteren nicht mehr kümmern.*

Im Folgenden können wir einen Fahrplan $f \in \mathscr{F}$ mit einer Kantenmenge $A_f \subset A$ identifizieren, da die Kanten Ankünfte und Abfahrten genau charakterisieren. Dabei muss darauf geachtet werden, dass dieser Teilgraph genau den Pfaden von σ nach τ in A^j jedes Zuges $j \in T$ entspricht. Dies motiviert die nächste Definition.

Definition 4.9. *Eine Kante* $a \in A$ *heißt genau dann* ***aktiv*** *bzgl.* $f \in \mathscr{F}$, *wenn sie in dem Fahrplan f benutzt wird, d.h.* $a \in A_f$. *Daraus ergibt sich*

$$A_f = \{a \in A : a \text{ aktiv bzgl. } f\}$$

mit $f \in \mathscr{F}$.

4.4 Kantenbewertung

Nachdem wir unseren Graph eingeführt haben, müssen wir diesen noch mit Kantenbewertungen versehen, die in diesem Fall Nutzen entsprechen.
Die Idee hierbei ist den Stretch und den Shift auf die einzelnen Kanten zu beziehen. Dabei fällt zunächst auf, dass der Shift nur von den Startkanten abhängt und wir somit jeder Startkante eine Bewertung in Höhe des idealen Nutzens, vermindert um die Bestrafung der verspäteten Abfahrt, zuweisen. Der Stretch tritt immer dann auf, wenn es zu „Verzögerungen“ kommt, welche immer in den Bahnhöfen oder auf einer Strecke auftreten. Also ist die Kantenbewertung für eine Bahnhofskante bzw. Streckenkante die Bestrafung, die durch eine „Verzögerung“ (einen Stretch) auf dieser Kante auftritt. Die Endkanten haben keinen Einfluss mehr auf den gesamten Stretch, weshalb wir hier Kantenbewertungen von 0 setzen. Summiert man anschließend auf einem Pfad alle Nutzen auf, erhält man genau den Nutzen, der im ersten Kapitel beschrieben wurde.

Wir beginnen wiederum mit den Startkanten und deren Nutzen. Hierbei wollen wir zunächst an die Funktion ϕ erinnern, die eine verspätete Abfahrt bestraft.

$$\phi_j : P \cup \{0\} \to \mathbb{R} \quad j \in T$$

Wir definieren nun wie oben beschrieben den Shift auf einer Startkante.

Definition 4.10. *Sei $j \in T$, $(\sigma, \nu) \in A_S^j$. Dann beschreibt die Funktion $\nu_j : W^{f_j} \to P \cup \{0\}$ den* ***Shift einer Startkante****.*

$$\nu_j(\nu) := |\theta(\nu) - \overline{D}_j|$$

Der Bezug zu dem Shift, der im ersten Kapitel eingeführt wurde, besteht darin, dass dies der Shift der aktiven Startkante ist.

$$V_j(f) = \sum_{(\sigma,\nu) \in A_f \cap A_S^j} \nu_j(\nu)$$

Damit lässt sich nun der Nutzen für solche Startkanten angeben,

$$p_{(\sigma,\nu)} := \Pi_j - \phi_j(\nu_j(\nu)), \quad j \in T, \quad (\sigma, \nu) \in A_1^j$$

wobei Π_j, wie im ersten Kapitel eingeführt, der ideale Nutzen des Zuges ist.

Im Folgenden wird der Stretch einer Bahnhofskante eingeführt.

Definition 4.11. *Sei $j \in T$, $i \in S^j \backslash \{f_j, l_j\}$, $(u, \nu) \in A_{B,i}^j$. Dann beschreibt die Funktion $\mu_{ij} : A_{B,i}^j \to P \cup \{0\}$ den* ***Stretch einer Bahnhofskante****.*

$$\mu_{ij}(u, \nu) := |\Delta(u, \nu) - \overline{S}_{ij}|$$

Nun geben wir mit Hilfe dieses Stretchs den Nutzen für eine Bahnhofskante an.

$$p_{(u,\nu)} := -\gamma_j \cdot \mu_{ij}(u, \nu), \quad j \in T, \quad i \in S^j \backslash \{f_j, l_j\}, \quad (u, \nu) \in A_{B,i}^j$$

wobei γ_j definiert ist wie im ersten Kapitel.
Wir definieren anschließend den Nutzen auf einer Streckenkante und müssen auch hierbei zunächst den Stretch auf einer solchen Kante definieren.

Definition 4.12. *Sei $j \in T$, $i \in S^j \backslash \{l_j\}$, $(\nu, u) \in A_{Str,i}^j$. Dann beschreibt die Funktion $\mu_{ij} : A_{Str,i}^j \to P \cup \{0\}$ den* ***Stretch einer Streckenkante****.*

$$\mu_{ij}(\nu, u) := |\Delta(\nu, u) - \overline{F}_{ij}|$$

Nun wird der Nutzen auf Streckenkanten äquivalent zu den Bahnhofskanten definiert.

$$p_{(\nu,u)} := -\gamma_j \cdot \mu_{ij}(\nu, u), \quad j \in T, \quad i \in S^j \backslash \{l_j\}, \quad (\nu, u) \in A_{Str,i}^j$$

Es lässt sich auch ein Zusammenhang zu dem Stretch, der im erste Kapitel definiert wurde, herstellen. Dieser besteht darin, dass die Stretchs auf den aktiven Bahnhofskanten und den aktiven Streckenkanten bezüglich des Zuges aufsummiert werden und man somit den gesamten Stretch erhält.

$$\mu_j(f) = \sum_{i \in S^j \backslash \{f_j, l_j\}} \sum_{a \in A_{B,i}^j \cap A_f} \mu_{ij}(a) + \sum_{i \in S^j \backslash \{l_j\}} \sum_{a \in A_{Str,i}^j \cap A_f} \mu_{ij}(a)$$

Wie oben erwähnt haben Endkanten den Nutzen 0.

$$p_{(u,\tau)} := 0, \quad j \in T, \quad (u, \tau) \in A_E^j$$

Zusammenfassend ist der Nutzen einer Kante wie folgt definiert

$$p_a := \begin{cases} \Pi_j - \phi_j(\nu_j(\nu)) & \text{wenn } a \in A_S^j \text{ mit } a = (\sigma, \nu) \\ -\gamma_j \cdot \mu_{ij}(a) & \text{wenn } a \in A_{B,i}^j \text{ mit } i \in S^j \backslash \{f_j, l_j\} \\ -\gamma_j \cdot \mu_{ij}(a) & \text{wenn } a \in A_{Str,i}^j \text{ mit } i \in S^j \backslash \{l_j\} \\ 0 & \text{wenn } a \in A_E^j \end{cases}$$

mit $j \in T$ und $a \in A^j$.
Der Nutzen eines Fahrplans ist nun dadurch gegeben, dass man die Nutzen der aktiven Kanten aufaddiert.

$$N(f, j) = \sum_{a \in A_f \cap A^j} p_a \text{ und } N_f = \sum_{a \in A_f} p_a$$

4.5 zulässige Pfade

In diesem Abschnitt setzen wir uns nun damit auseinander, wie eine Lösung in diesem Graph gefunden werden kann. Wir gehen jedoch zunächst noch einmal auf die Streckenkapazitätsbeschränkungen ein.

Definition 4.13. *Sei $j,k \in T$, $i \in (S_j \backslash \{l_j\}) \cap (S^k \backslash \{l_k\})$ und $(\nu_1, u_1) \in A^j_{Str,i}$, $(\nu_2, u_2) \in A^k_{Str,i}$. Dann nennen wir das Paar $(\nu_1, u_1), (\nu_2, u_2)$ **inkompatibel**, wenn mindestens eine der folgenden drei Bedingungen erfüllt ist:*

(i) $\nu_1 \preceq \nu_2$ *und* $\Delta(\nu_1, \nu_2) < d_i$ *(Abfahrt zu eng)*

(ii) $u_1 \preceq u_2$ *und* $\Delta(u_1, u_2) < a_{i+1}$ *(Ankunft zu eng)*

(iii) $\nu_1 \preceq \nu_2$ *und* $u_2 \preceq u_1$ *(Überholvorgang außerhalb eines Bahnhofs)*

*Ansonsten nennen wir das Paar **kompatibel**.*

Damit lässt sich auch eine Zulässigkeitsbedingung für einen Fahrplan angeben, die zusichert, dass der Plan unseren Wünschen entspricht.

Definition 4.14. *Ein Fahrplan $f \in \mathscr{F}$ heißt **zulässig**, falls alle aktiven Kanten bzgl. f paarweise kompatibel sind, d.h. für alle $a_1, a_2 \in A_f$ gilt a_1 und a_2 sind kompatibel.*

Definition 4.15. *Sei $j \in T$. Sei $u_1, \ldots, u_n \in V$ mit $n \in \mathbb{N}$. Dann heißt*

$$P^j := \{(\sigma, u_1), (u_1, u_2), \ldots, (u_{n-1}, u_n), (u_n, \tau)\}$$

*mit $(\sigma, u_1), (u_n, \tau), (u_i, u_{i+1}) \in A^j (i \in \{1, \ldots, n-1\})$ **Pfad im Graph G** in Bezug auf Zug j.*

Notation 4.4. *Wir nennen die **Menge aller Pfade** in einem Graph G bzgl. eines Zuges j $\mathscr{P}^j(G)$. Daraus ergibt sich auch die Menge aller Pfade eines Graphen G durch die Vereinigung.*

$$\mathscr{P}(G) := \bigcup_{j \in T} \mathscr{P}^j(G)$$

Oder kurz: $\mathscr{P}$

Da die Knoten eines Pfades an einem späteren Punkt benötigt werden, führen wir nun eine Menge ein, welche genau die von einem Pfad betroffenen Knoten enthält.

Notation 4.5. *Sei $P \in \mathscr{P}$ ein Pfad. Dann nennen wir $K(P)$ die **Knotenmenge des Pfades P**.*

$$K(P) := \{\nu \in V \backslash \{\sigma, \tau\} : \exists w \in V \text{ mit } (\nu, w) \in P\} \subseteq V^j$$

Der nächste Satz charakterisiert die Bedeutung der Pfade in unserem Problem.

Satz 4.1. *Sei $j \in T$. Jeder zulässige Pfad P^j in Bezug auf j stellt einen Fahrplan für den Zug j dar.*

Beweis. Aus obiger Bemerkung folgt, dass durch die Konstruktion des Graphen bereits alle Beschränkungen eingehalten werden. Da unsere zulässigen Pfade in σ starten und man von dort aus nur in einen Knoten, der zur Abfahrt in f_j gehört, gelangt, ist die Abfahrt im richtigen Bahnhof gesichert. Dazwischen kommt man durch Kanten in A^j nur von Ankunft zu Abfahrt in einem Bahnhof bzw. von der Abfahrt in einem Bahnhof zur Ankunft im nächsten Bahnhof. Demnach müssen alle Bahnhöfe besucht werden. Ist ein Halt nicht vorgesehen, entspricht der Ankunftszeitpunkt dem Abfahrtszeitpunkt in dem Bahnhof. Sind wir im letzten Bahnhof l_j angekommen existieren nur Kanten nach τ. Damit ist gesichert, dass auch alle gewünschten Bahnhöfe besucht werden und es ergibt sich ein zulässiger Fahrplan für den Zug. □

Um einem Pfad eine Güte zuzuweisen, lässt sich ein Nutzen für einen Pfad angeben, der sich durch die aufsummierten Nutzen der im Pfad enthaltenen Kanten ergibt.

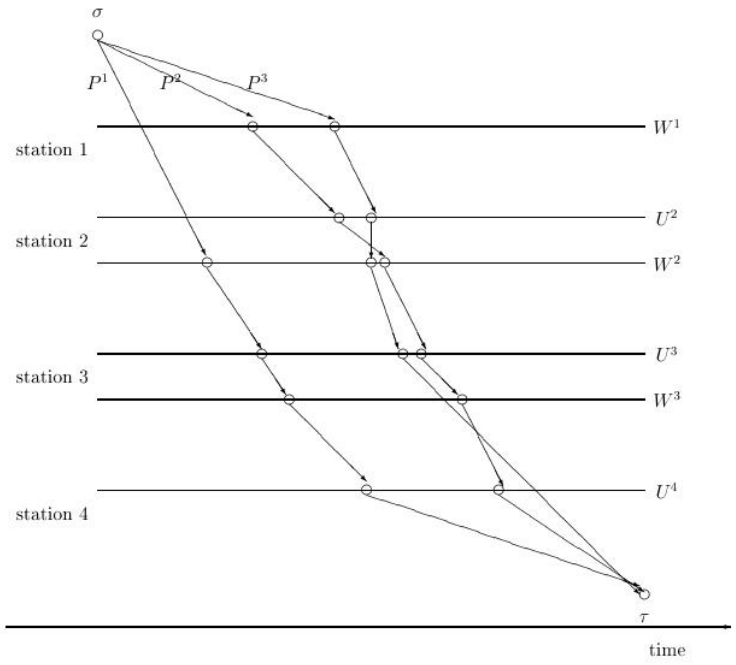
$$N : \mathscr{P}(G) \to \mathbb{R} \text{ mit } N(P) := \sum_{a \in P} p_a$$

Beispiel. *Wir geben nun ein Beispiel für Pfade in einem Graphen an. Dabei haben wir folgende Parameter:*

- *4 Bahnhöfe ($s = 4$)*
- *3 Züge ($t = 3$)*

- $f_1 = 2$ *und* $l_1 = 4$
- $f_2 = 1$ *und* $l_2 = 4$
- $f_3 = 1$ *und* $l_3 = 3$

Dann ergeben sich folgende drei Pfade im erstellten Graphen:



5 linear ganzzahlige Programme

Nachdem wir eine Modellierung in Form eines Flussgraphen gefunden haben, entwickeln wir nun ein Lineares Program. Dieses stellt sich jedoch als ungünstig heraus und wir müssen es somit etwas erweitern, um anschließend einen Lösungsalgorithmus angeben zu können.

5.1 Erstes ganzzahlig lineares Programm

In diesem Kapitel entwickeln wir ein erstes lineares Programm des TTP. Dazu führen wir eine Entscheidungsvariable ein. Diese Variable soll angeben, ob eine Kante aktiv ist oder nicht. Sie ist der zentrale Punkt dieser Modellierung des TTP.

$$x_a = \begin{cases} 1 & ; \text{ Kante a ist in der Lösung enthalten} \\ 0 & ; \text{ sonst} \end{cases}$$

mit $a \in A$.

Man kann mit Hilfe dieser Variable eine aktive Kante beschreiben.

Bemerkung. *Eine Kante $a \in A$ ist genau dann aktiv, wenn $x_a = 1$ gilt.*

Bemerkung. *Sei $j \in T$, $a \in A^j$ und $x_a = 1$. Dann enthält der Pfad des Zuges j in der Lösung die Kante a.*

Die Idee der LP-Modellierung ist, dass man für jeden Zug, falls er nicht ausfällt, einen Fluss der Größe 1 garantiert, der von der Quelle σ zur Senke τ fließt, wobei die Variable x_a angibt, ob auf der Kante der Fluss 1 oder 0 ist. Man kann sich also für jeden Zug einen eigenen Graph mit Knotenmenge V und Kantenmenge A^j vorstellen. In diesem Graph sucht man einen nutzenoptimalen Pfad von σ nach τ. Zusätzlich muss noch darauf geachtet werden, dass alle aktiven Kanten zueinander kompatibel sind. Dies entspricht der Grundidee eines Multi-Commodity-Flows[4].

Da wir zur Flussbetrachtung eingehende und ausgehende Kanten zugspezifisch betrachten, führen wir die folgenden zwei Mengen ein

$$\delta_j^+(v) := \{a \in A^j : a = (v, \alpha); \alpha \in V\}$$

mit $j \in T$ und $v \in V$.
Diese Menge beschreibt die Kanten des Zuges j, die den Knoten v **verlassen**.

$$\delta_j^-(v) := \{a \in A^j : a = (\alpha, v); \alpha \in V\}$$

mit $j \in T$ und $v \in V$.
Diese Menge beschreibt die Kanten des Zuges j, die in den Knoten v **eintreten**.

Wir wollen nun den oben genannten Fluss modelieren. Dazu muss zunächst gesichert sein, dass maximal eine Flusseinheit aus der Quelle herausfließt. Dies sichern die folgenden Ungleichungen.

$$\sum_{a \in \delta_j^+(\sigma)} x_a \leq 1$$

für alle $j \in T$

Dieser ausgehende Fluss muss sich bis zur Senke fortpflanzen. Es muss also eine Flusserhaltung gesichert sein.

$$\sum_{a \in \delta_j^-(v)} x_a = \sum_{a \in \delta_j^+(v)} x_a$$

für alle $j \in T$, $v \in V \backslash \{\sigma, \tau\}$

Mit diesen zwei Nebenbedingungen lässt sich nun der oben genannte Fluss garantieren. Im nächsten Schritt muss die Inkompatibilität ausgeschlossen werden. Dazu benötigen wir ein Mengensystem, welches aus maximal inkompatiblen Kantenmengen besteht. Dies motiviert die folgende Definition.

Definition 5.1. *Eine Menge $C \subset A$ heißt **maximal**, falls man keine weitere Kante hinzufügen kann, die zu allen Kanten in der Menge inkompatibel ist.*

Das Mengensystem lässt sich nun wie folgt angeben:

$$\mathscr{C} := \{C \subset A : \text{ Elemente von C sind paarweise inkompatibel und C maximal}\}$$

Wenn wir uns eine Menge aus diesem Mengensystem nehmen, darf jeweils nur eine Kante aus dieser aktiv sein, da ansonsten zwei inkompatible Kanten aktiv sind, was einer Streckenkapazitätsbeschränkung widerspricht. Dies sichert folgende Nebenbedingung.

$$\sum_{a \in C} x_a \leq 1$$

für alle $C \in \mathscr{C}$

Man kann die oben erhaltenen Erkenntnisse zusammen in einem LP sammeln.

Lineares Programm 5.1.

$$\max \sum_{j \in T} \sum_{a \in A^j} p_a \cdot x_a \tag{1}$$

s.t.:

Flussbedingungen:

$$\sum_{a \in \delta_j^+(\sigma)} x_a \leq 1 \tag{2}$$

für alle $j \in T$

$$\sum_{a \in \delta_j^-(v)} x_a = \sum_{a \in \delta_j^+(v)} x_a \tag{3}$$

für alle $j \in T$, $v \in V \backslash \{\sigma, \tau\}$

Cliquenbedingung:

$$\sum_{a \in C} x_a \leq 1 \tag{4}$$

für alle $C \in \mathscr{C}$

Variablendefinition:

$$x_a \geq 0, x_a \in \mathbb{Z} \tag{5}$$

für alle $a \in A$

Die folgende Bemerkung garantiert uns die Flusserhaltung.

Bemerkung. *Seien die Nebenbedingungen (2)-(5) erfüllt. Dann gilt*

$$\sum_{a \in \delta_j^+(v)} x_a \leq 1 \text{ und } \sum_{a \in \delta_j^-(v)} x_a \leq 1$$

für alle $j \in T$ *und alle* $v \in V \backslash \{\sigma, \tau\}$. *Desweiteren gilt*

$$\sum_{a \in \delta_j^+(\sigma)} x_a \leq 1 \text{ und } \sum_{a \in \delta_j^-(\tau)} x_a \leq 1$$

Wir wollten unsere Entscheidungsvariable x_a so modellieren, dass sie nur die Werte 0 oder 1 annimmt. Dass dies gelungen ist, zeigt der folgende Satz.

Satz 5.1. *Seien die Nebenbedingung (2),(3),(5) erfüllt. Dann gilt*

$$x_a \in \{0,1\}$$

mit $a \in A$.

Beweis. Sei $a_0 \in A$. Daraus folgt, dass es ein $v \in V\backslash\{\tau\}$, ein $w \in V\backslash\{\sigma\}$ und ein $j \in T$ gibt, so dass

$$a_0 = (v, w) \text{ und } a_0 \in A^j$$

gilt. Damit ergibt sich

$$a_0 \in \delta_j^+(v)$$

Mit Nebenbedingung (5) ($x_a \geq 0$) und der Flusserhaltung folgt nun:

$$x_{a_0} \leq \sum_{a \in \delta_j^+(v)} x_a \leq 1$$

Da $x_{a_0} \in \mathbb{Z}$ und $x_{a_0} \geq 0$ (5), folgt direkt $x_{a_0} \in \{0,1\}$. □

5.2 Lösungsansatz erstes ganzzahlig lineares Programm

Da wir eine sehr große Anzahl an Nebenbedingungen und Variablen haben, lässt sich kein effizienter Algorithmus entwickeln, da dieser sehr große Komplexität hätte. In solchen Situationen hat sich die Methode der Lagrange-Relaxation verbunden mit der Subgradientenoptimierung als sinnvoll herausgestellt [5]. Dieses Verfahren liefert uns dann eine heuristische Lösung.

Wir relaxieren also die Cliquennebenbedingung (4), da diese exponentiell viele Nebenbedinungen generiert. Dazu führen wir für jedes Element aus $\mathscr{C}$ einen Lagrangemultiplikator ein.

Definition 5.2. *Sei* $C \in \mathscr{C}$. *Dann ist* $\lambda_C \geq 0$ *der* ***Lagrangemultiplikator*** *verbunden mit der Nebenbedingung die zu* C *gehört.*

Es ergibt sich nun unsere relaxierte Zielfunktion

$$\max \sum_{j \in T} \sum_{a \in A^j} p_a \cdot x_a + \sum_{C \in \mathscr{C}} \lambda_C \cdot (1 - \sum_{a \in C} x_a)$$

Durch die Relaxation ändert sich der Nutzen einer Kante. Dazu führen wir den Lagrangenutzen ein.

Definition 5.3. *Sei* $a \in A$. *Dann nennen wir*

$$\overline{p}_a := p_a - \sum_{C \in \mathscr{C}_a} \lambda_C$$

den ***Lagrangenutzen der Kante a****, wobei die Menge* $\mathscr{C}_a$ *wie folgt definiert ist*

$$\mathscr{C}_a := \{C \in \mathscr{C} : a \in C\}$$

Nun können wir unsere Zielfunktion mit Hilfe des Lagrangenutzens angeben.

$$\begin{aligned}
& \sum_{j \in T} \sum_{a \in A^j} p_a \cdot x_a + \sum_{C \in \mathscr{C}} \lambda_C \cdot (1 - \sum_{a \in C} x_a) \\
= & \sum_{j \in T} \sum_{a \in A^j} p_a \cdot x_a - \sum_{C \in \mathscr{C}} \sum_{a \in C} \lambda_C \cdot x_a + \sum_{C \in \mathscr{C}} \lambda_C \\
= & \sum_{j \in T} \sum_{a \in A^j} p_a \cdot x_a - \sum_{j \in T} \sum_{a \in A^j} \sum_{C \in \mathscr{C}_a} \lambda_C \cdot x_a + \sum_{C \in \mathscr{C}} \lambda_C \\
= & \sum_{j \in T} \sum_{a \in A^j} (p_a - \sum_{C \in \mathscr{C}_a} \lambda_C) \cdot x_a + \sum_{C \in \mathscr{C}} \lambda_C \\
= & \sum_{j \in T} \sum_{a \in A^j} \overline{p}_a \cdot x_a + \sum_{C \in \mathscr{C}} \lambda_C
\end{aligned}$$

Dabei ist der Knackpunkt zu erkennen, dass

$$\sum_{C\in\mathscr{C}}\sum_{a\in C}\lambda_C\cdot x_a=\sum_{j\in T}\sum_{a\in A^j}\sum_{C\in\mathscr{C}_a}\lambda_C\cdot x_a$$

gilt. Dies sieht man ein, in dem man auf der linken Seite über alle Elemente in den Cliquen der Menge $\mathscr{C}$ aufsummiert und auf der rechten Seite nimmt man alle Kanten und summiert über die Cliquen, in denen die Kante enthalten ist, auf. Damit sind beide Seiten gleich.

Damit ergibt sich ein neues ganzzahlig lineares Programm, welches wir durch die Lagrange-Relaxation erhalten haben.

Lineares Programm 5.2.

$$\max\sum_{j\in T}\sum_{a\in A^j}\overline{p}_a\cdot x_a+\sum_{C\in\mathscr{C}}\lambda_C \tag{6}$$

s.t.:

Flussbedingungen:

$$\sum_{a\in\delta_j^+(\sigma)}x_a\leq 1 \tag{7}$$

für alle $j\in T$

$$\sum_{a\in\delta_j^-(v)}x_a=\sum_{a\in\delta_j^+(v)}x_a \tag{8}$$

für alle $j\in T,\ v\in V\backslash\{\sigma,\tau\}$

Variablendefinition:

$$x_a\geq 0, x_a\in\mathbb{Z} \tag{9}$$

für alle $a\in A$

Wir haben nun ein klassisches Multi-Commodity-Flow Problem [4] mit einer Nachfrage von jeweils 1, da ja nur ein Pfad für jeden Zug gesucht werden muss. Man kann für eine beliebige Wahl der λ_c's für jeden Zug einen Pfad mit maximalem Lagrangenutzen suchen. Dazu gibt es Standard-längste-Pfade Algorithmen in azyklischen Graphen. Diese haben eine Laufzeit von $O(|A^j|)$ für jeden Zug.

Man kann nun näherungsweise optimale Multiplikatoren durch Subgradientenoptimierung ermitteln und bekommt dadurch eine näherungsweise optimale Lösung.

Damit hätten wir einen Lösungsalgorithmus gefunden. Das Problem ist jedoch, dass die Anzahl der Variablen sehr groß ist und für jeden von diesen der Lagrangenutzen berechnet werden muss, d.h. wir müssen für jede Kante die dazugehörigen Cliquen entwickeln. Diese Berechnung ist bei Realinstanzen auf Grund der Größe unmöglich zu realisieren. Aus diesem Grund müssen wir uns einen neuen Lösungsansatz suchen. Dazu erweitern wir das Modell.

5.3 Erweiterung des Modells

Die Idee ist nun, die Flussbedingung unverändert zu lassen, jedoch zur Sicherung der Inkompatibilität, Variablen einzuführen, die sich nicht auf die Kanten sondern auf die Knoten beziehen. Dies hat den Vorteil, dass es weniger Knoten als Kanten gibt und somit die Laufzeit verbessert wird.
Dazu führen wir zunächst eine Menge der für einen Zug „interessanten“ Knoten ein.

$$V^j:=\{v\in V\backslash\{\sigma,\tau\}:\delta_j^+(v)\neq\emptyset,\delta_j^-(v)\neq\emptyset\}\quad j\in T$$

Diese Menge beschreibt die Menge der Knoten, die wir mit dem Zug j verbinden und die auch als mögliche Durchfahrtsknoten in Betracht kommen. Es würde keinen Sinn machen einen Knoten zu betrachten, der nicht erreicht wird oder in dem man feststeckt.

Bemerkung. *Aus der Definition von* V^j *folgt* $V^j\subseteq V\backslash\{\tau,\sigma\}$.

Notation 5.1. *Ein Knoten* $v\in V$ *heißt* ***besucht*** *vom Zug* $j\in T$ *bzgl.* $f\in\mathscr{F}$*, falls* $(u,v),(v,w)\in A_f\cap A^j$ *existieren, d.h.* $x_{(u,v)}=x_{(v,w)}=1$.

Die Variable x_a lassen wir unverändert und definieren sie wie im vorherigen Kapitel, da sie für die Flussbedingungen benötigt wird. Unsere erste neue Variable ist z_{jv}, welche angibt, ob ein Knoten von einem Zug besucht wird. Wir gehen auch hierbei von x_a aus.

$$z_{jv} = \sum_{a \in \delta_j^-(v)} x_a \text{ mit } j \in T,\ v \in V^j$$

Bemerkung. *Sei $j \in T$ und $v \in V^j$. Dann gilt $z_{jv} \in \{0,1\}$.*

Beweis. Sei $j \in T$,$v \in V^j$ und $a \in \delta_j^-(v)$. Aus Satz 5.1 folgt $x_a \in \{0,1\}$. Daraus ergibt sich

$$\sum_{a \in \delta_j^-(v)} x_a \in \mathbb{N}_0$$

Da $v \in V^j \subseteq V \backslash \{\tau, \sigma\}$ gilt, liefert die Flusserhaltung $\sum_{a \in \delta_j^-(v)} x_a \leq 1$. Daraus folgt direkt

$$z_{jv} = \sum_{a \in \delta_j^-(v)} x_a \in \{0,1\}$$

□

Dies bestätigt unsere Modellierung der Variablen und man kann sie wie folgt charakterisieren.

Satz 5.2. *Sei $j \in T$ und $v \in V^j$. Dann gilt*

$$z_{jv} = \begin{cases} 1 & ;\ \textit{Knoten } v \textit{ wird von Zug } j \textit{ besucht} \\ 0 & ;\ \textit{sonst} \end{cases}$$

Beweis. Sei $j \in T$ und $v \in V^j$.
Fall 1: der Pfad von Zug j besucht Knoten v
Daraus folgt, dass ein $a_0 = (\alpha, v) \in A^j$ existiert mit $x_{a_0} = 1$. Aus der Form von a_0 folgt $a_0 \in \delta_j^-(v)$. Nun ergibt sich direkt

$$z_{jv} = \sum_{a \in \delta_j^-(v)} x_a \geq x_{a_0} = 1$$

Dies ergibt wiederum zusammen mit der vorherigen Bemerkung $z_{jv} = 1$.

Fall2: der Pfad von Zug j besucht nicht Knoten v
Sei $a \in \delta_j^-(v)$. Da der Knoten v von Zug j nicht besucht wird, gilt $x_a = 0$. Daraus ergibt sich

$$z_{jv} = \sum_{a \in \delta_j^-(v)} x_a = \sum_{a \in \delta_j^-(v)} 0 = 0$$

□

Ausgehend von z_{jv} definieren wir uns nun eine zweite neue Variable y_v, die angeben soll, ob der Knoten v überhaupt besucht wird. Dabei gehen wir wie folgt vor. Wir summieren über unsere Variable z_{jv} auf und erhalten das gewünschte Ergebnis. Da wir nicht davon ausgehen können, dass ein Knoten $v \in V$ für jeden Zug $j \in T$ in V^j liegt, motiviert dies die Definition der folgenden Indexmenge.

$$I_v := \{j \in T : v \in V^j\} \subseteq T, \quad v \in V$$

Diese Menge gibt an, welche Züge in Frage kommen über den Knoten v zu fahren, da sie eingehende und ausgehende Kanten in bzw. aus diesem Knoten haben.

Wir geben nun die gewünschte Variable an und überlegen uns anschließend welche Werte sie wann animmt.

$$y_v := \sum_{j \in I_v} z_{jv}, \quad v \in V$$

Wir untersuchen zunächst die neue Indexmenge, um anschließend y_v charakterisieren zu können.

Lemma 5.1. *Sei $v \in V$. Aus $I_v = \emptyset$ folgt, dass der Knoten v von keinem Zug besucht wird.*

Beweis. Sei $I_v = \emptyset$. Daraus ergibt sich, dass kein $j \in T$ existiert mit $v \in V^j$. Die Definition von V^j liefert

$$\delta_j^+(v) = \emptyset \text{ oder } \delta_j^-(v) = \emptyset \quad \forall j \in T$$

Das ergibt, dass man entweder nicht in den Knoten v hinein kommt oder nicht aus dem Knoten heraus kommt. Also wird der Knoten v von keinem Zug besucht. □

Bemerkung. *Sei $v \in V$. Wenn $I_v = \emptyset$ ist, folgt daraus auch direkt, dass $y_v = 0$, da über nichts aufsummiert wird. Dies macht nach Lemma 5.1 auch Sinn, da dann der Knoten auch von keinem Zug besucht wird.*

Lemma 5.2. *Sei $v \in V$. Aus $I_v \neq \emptyset$ und $\sum_{j \in I_v} z_{jv} = 0$ folgt, dass der Knoten v von keinem Zug besucht wird.*

Beweis. Sei $v \in V$, $I_v \neq \emptyset$ und $\sum_{j \in I_v} z_{jv} = 0$.
Da $z_{jv} \in \{0, 1\}$ muss $z_{jv} = 0$ für alle $j \in I_v$ gelten.
Daraus folgt bereits, dass der Knoten v von keinem Zug aus I_v besucht wird. Wir wenden uns nun den restlichen Zügen zu.
Sei $j \in T \backslash I_v$. Dann folgt daraus $v \notin V^j$ und somit $\delta_j^+(v) = \emptyset$ oder $\delta_j^-(v) = \emptyset$.
Sei OBdA $\delta_j^-(v) = \emptyset$.

$$0 = \sum_{a \in \delta_j^-(v)} x_a = \sum_{a \in \delta_j^+(v)} x_a$$

Daraus ergibt sich $x_a = 0$ für alle $a \in \delta_j^+(v)$ und somit wird der Knoten auch von den restlichen Zügen nicht besucht, folglich ist die Behauptung bewiesen. □

Nun lässt sich mit diesen Lemmata der eigentliche Satz beweisen.

Satz 5.3. *Sei $v \in V$. Der Knoten v wird genau dann von keinem Zug besucht, wenn $y_v = 0$ gilt.*

Beweis. Sei $v \in V$

„⇐“ Sei $y_v = 0$.
Dann gilt entweder $I_v = \emptyset$ oder $I_v \neq \emptyset$ und $\sum_{j \in I_v} z_{jv} = 0$. Nach Lemma 5.1 und Lemma 5.2 folgt in beiden Fällen, dass der Knoten v von keinem Zug besucht wird.

„⇒“ Annahme: Der Knoten v wird von keinem Zug besucht.
Dann gilt

$$z_{jv} = 0 \quad \forall j \in I_v$$

da in I_v die einzigen Züge sind die v überhaupt ansteuern können.
Daraus folgt nun direkt, dass

$$y_v = \sum_{j \in I_v} z_{jv} = 0$$

Gilt $I_v = \emptyset$ folgt die Behauptung direkt, da wir über die leere Menge aufsummieren.

□

Bemerkung. *Sei $v \in V$. Man kann im Moment nur sagen, dass y_v angibt wieviele Züge den Knoten v besuchen. Im weiteren Verlauf können wir jedoch zeigen, dass $y_v \in \{0, 1\}$ gilt.*

Wir haben noch eine kleine Schwierigkeit mit der Wohldefinitheit von y_v. Da σ und τ in keinem V^j enthalten sind, folgt daraus, dass $I_\tau = I_\sigma = \emptyset$ und somit $y_\tau = y_\sigma = 0$ gilt. Jedoch ist dies nicht weiter schlimm, da wir diesen Knoten eine Sonderstellung zuräumen, da in ihnen ohnehin mehrere Züge zur gleichen „Zeit“ ankommen.

Mit Hilfe dieser Variablen können wir nun die Streckenkapazitätsbedingungen modellieren. Wir betrachten zunächst die gleichzeitige Ein- bzw. Ausfahrt zweier Züge. Hierbei ist es hilfreich jeweils eine Menge der Knoten in der Umgebung um

einen Ankunfts- bzw. Abfahrtsknoten zu haben und in diesen Mengen anschließend nur einen Knoten zuzulassen. Dies motiviert die folgenden zwei Mengendefinitionen.

$$M_{a_i}(u) := \{w \in U^i : u \preceq w; \Delta(u,w) < a_i\} \subseteq V \quad i \in S\backslash\{1\};\ u \in U^i$$

$$N_{d_i}(u) := \{w \in W^i : u \preceq w; \Delta(u,w) < d_i\} \subseteq V \quad i \in S\backslash\{s\};\ u \in W^i$$

Unsere Nebenbedingung soll nun garantieren, dass in jeder dieser Mengen maximal ein Knoten besucht wird, da es ansonsten zu einer zu engen Ankunft bzw. Abfahrt zweier Züge kommt. Dies lässt sich mit y_v modellieren.

$$\sum_{w \in M_{a_i}(u)} y_w \leq 1 \quad i \in S\backslash\{1\};\ u \in U^i$$

$$\sum_{w \in N_{d_i}(u)} y_w \leq 1 \quad i \in S\backslash\{s\};\ u \in W^i$$

An vorheriger Stelle hatten wir gesagt, dass man zeigen kann, dass für alle $v \in V$ $y_v \in \{0,1\}$ gilt. Diese Aussage lässt sich nun mit Hilfe der Streckenkapazitätsbeschränkungen beweisen, was wir in den nächsten zwei Lemmata und dem darauf aufbauenden Satz angehen wollen.

Lemma 5.3.

(i) *Sei $i \in S\backslash\{1\}$ und $u \in U^i$. Dann gilt $u \in M_{a_i}(u)$.*

(ii) *Sei $j \in S\backslash\{s\}$ und $w \in W^i$. Dann gilt $w \in N_{d_j}(w)$.*

Beweis.

(i) Sei $i \in S\backslash\{1\}$ und $u \in U^i$.
Es gilt $u \preceq u$, da „$\preceq$“ reflexiv ist, und $\Delta(u,u) = 0 < a_i$.
Daraus ergibt sich schon $u \in M_{a_i}(u)$.

(ii) folgt analog zu Fall (i).

□

Lemma 5.4. *Sei $v \in V\backslash\{\sigma,\tau\}$. Dann gilt $y_v \in \{0,1\}$.*

Beweis. Sei $v \in V\backslash\{\sigma,\tau\}$.
Aus der Definition von V folgt, dass ein $i \in S$ existiert, so dass $v \in U^i$ oder $v \in W^i$ gilt. Es ergibt sich nun mit Lemma 5.3

$$v \in M_{a_i}(v) \text{ oder } v \in N_{d_i}(v)$$

Es ergibt sich direkt aus den oben definierten Nebenbedingungen

$$y_v \leq \sum_{w \in M_{a_i}(v)} y_w \leq 1 \text{ oder } y_v \leq \sum_{w \in N_{d_i}(v)} y_w \leq 1$$

Im nächsten Schritt sieht man nun, dass $y_v \in \mathbb{N}_0$, also auch $y_v \geq 0$, da $z_{jv} \in \{0,1\}$. Damit gilt direkt $y_v \in \{0,1\}$. □

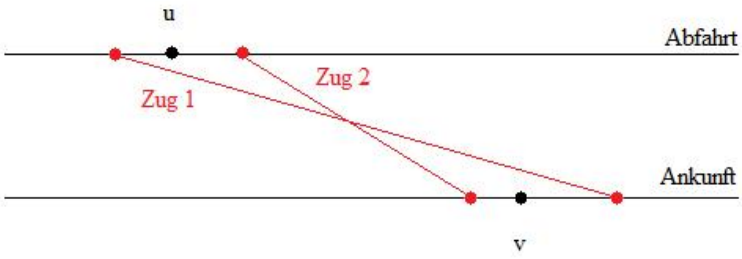
Jetzt lässt sich eine äquivalente Definition von y_v angeben, die uns eine etwas zugänglichere Betrachtungsweise erlaubt.

Satz 5.4. *Sei $v \in V\backslash\{\sigma,\tau\}$. Dann gilt*

$$y_v = \begin{cases} 1 & ; \textit{Knoten v wird von genau einem Zug besucht} \\ 0 & ; \textit{Knoten v wird von keinem Zug besucht} \end{cases}$$

Beweis. Sei $v \in V\backslash\{\sigma,\tau\}$.
Das $y_v = 0$ genau dann gilt, wenn der Knoten v von keinem Zug besucht wird, liefert uns der Satz 5.3. Also müssen wir noch zeigen, dass $y_v = 1$ genau dann gilt, wenn ein Zug existiert, der den Knoten v besucht.
Sei also $y_v = 1$. Dies ist äquivalent dazu, dass $y_v \neq 0$. Da wir im ersten Teil schon die Äquivalenz gezeigt haben, ist dies wiederum äquivalent dazu, dass der Knoten v (nicht nicht) besucht wird. Somit ist auch hier die Äquivalenz gezeigt.
Lemma 5.4 liefert uns abschließend, dass y_v nur diese beiden Werte annehmen kann. □

Dies zeigt uns nun, dass wir auch die Variable y_v so definiert haben, wie wir es uns vorgestellt hatten. Wir müssen uns jetzt noch in einem letzten Schritt dem „Überholverbot“ zuwenden. Dies können wir in sofern abstrahieren, dass wir für zwei Knoten, wobei der erste Abfahrts- und der zweite Ankunftsknoten ist, und zwei voneinander verschiedene Züge es nicht zulassen, dass ein Zug vor dem ersten Knoten losfährt und nach dem zweiten Knoten ankommt und der andere Zug nach dem ersten Knoten losfährt und vor dem zweiten Knoten ankommt, also anschaulich



Dies müssen wir nun in eine mathematische Modellierung fassen. Dazu definieren wir uns zunächst vier Mengen, die uns die benötigten Knoten sammeln.

$$De_j^i(v) := \{w \in W^i \cap V^j : w \preceq v\}$$

$$Dl_j^i(v) := \{w \in W^i \cap V^j : v \preceq w\}$$

$$Ae_j^i(u) := \{w \in U^i \cap V^j : w \preceq u\}$$

$$Al_j^i(u) := \{w \in U^i \cap V^j : u \preceq w\}$$

mit $i \in S\backslash\{\sigma\}, v \in W^i, u \in U^i$ und $j \in T$.

Die Menge $De_j^i(v)$ (**D**eparture **e**arlier) beschreibt alle Abfahrtsknoten in W^i, die zeitlich früher liegen als der Knoten v. Entsprechend beschreibt die Menge $Dl_j^i(v)$ (**D**eparture **l**ater) alle Abfahrtsknoten, die zeitlich hinter dem Knoten v liegen. Genauso sind auch die Mengen $Ae_j^i(u)$ (**A**rrival **e**arlier) und $Al_j^i(u)$ (**A**rrival **l**ater) definiert, nur in Bezug auf Ankunftsknoten.

Bemerkung. *Man kann einsehen, dass*

$$\sum_{w \in W^i} z_{jw}, \sum_{w \in U^i} z_{jw} \in \{0, 1\}$$

gilt. Dies kann man mit Hilfe des Satzes 4.1 sehen, da dieser uns einen korrekten Fahrplan zusichert. Und dieser garantiert auch, dass ein Zug nicht in einem Bahnhof mehr als einmal ein- bzw. ausfährt. Somit müssen diese Summen 0 oder 1 sein.

Damit können wir unsere angestrebte Nebenbedingung formulieren.

$$\sum_{w \in De_j^i(v)} z_{jw} + \sum_{w \in Al_j^{i+1}(u)} z_{jw} + \sum_{w \in Dl_k^i(v)} z_{kw} + \sum_{w \in Ae_k^{i+1}(u)} z_{kw} \leq 3$$

mit $i \in S\backslash\{s\}, v \in W^i, u \in U^{i+1}, j, k \in T$ wobei $j \neq k$.

Diese Nebenbedingung garantiert nun, dass maximal drei von den in der Abbildung roten Knoten angefahren werden können, was wiederum mit sich bringt, dass Überholvorgänge ausgeschlossen sind, da die einzelnen Summen 0 oder 1 ergeben. Da wir somit unsere Modellierung abgeschlossen haben, können wir das entstandene lineare Programm angeben.

Lineares Programm 5.3.

$$\max \sum_{j \in T} \sum_{a \in A^j} p_a \cdot x_a \tag{10}$$

s.t.:

Flussbeschreibung:

$$\sum_{a \in \delta_j^+(\sigma)} x_a \leq 1 \tag{11}$$

für alle $j \in T$

$$\sum_{a \in \delta_j^-(v)} x_a = \sum_{a \in \delta_j^+(v)} x_a \tag{12}$$

für alle $j \in T,\ v \in V \backslash \{\sigma, \tau\}$

Streckenbedingungen:

$$\sum_{w \in M_{a_i}(u)} y_w \leq 1 \tag{13}$$

für alle $i \in S \backslash \{1\};\ u \in U^i$

$$\sum_{w \in N_{d_i}(u)} y_w \leq 1 \tag{14}$$

für alle $i \in S \backslash \{s\};\ u \in W^i$

$$\sum_{w \in De_j^i(v)} z_{jw} + \sum_{w \in Al_j^{i+1}(u)} z_{jw} + \sum_{w \in Dl_k^i(v)} z_{kw} + \sum_{w \in Ae_k^{i+1}(u)} z_{kw} \leq 3 \tag{15}$$

für alle $i \in S \backslash \{s\}, v \in W^i, u \in U^{i+1}, j, k \in T$ *wobei* $j \neq k$

Variablendefinitionen:

$$z_{jv} = \sum_{a \in \delta_j^-(v)} x_a \tag{16}$$

für alle $j \in T,\ v \in V^j$

$$y_v = \sum_{j \in I_v} z_{jv} \tag{17}$$

für alle $v \in V$

$$x_a \geq 0,\ x_a \in \mathbb{Z} \tag{18}$$

für alle $a \in A$

Notation 5.2. *Durch die Modellierung des LP's existiert in einer mit dem Lösungsfahrplan* $f \in \mathscr{F}$ *verbundenen Kantenmenge* A_f *für jeden Zug* $j \in T$ *ein Pfad. Dieser kann auch leer sein, falls der Zug ausfällt. Wir nennen diesen Pfad* P_f^j.

Der Vorteil wird sich nun in der Entwicklung eines Lösungsalgorithmus zeigen, den wir im nächsten Kapitel entwickeln wollen.

Auch bei der Lösungsfindung für das lineare Programm 5.3 beginnen wir mit einer Lagrange-Relaxation, wobei wir die Streckennebenbedingungen relaxieren. Dies ist hier einfacher, da wir keine Cliquennebenbedinungen mehr haben, die sehr groß werden. Wir beginnen zunächst mit ein paar Notationen, die uns die Darstellung etwas vereinfachen, da wir die Nebenbedingungen, die relaxiert werden sollen, in Mengen sammeln. Wir nennen $\mathscr{H}_d$ die Menge der Nebenbedingungen, die die korrekten Abfahrten garantieren. Äquivalent für die Ankunft definieren wir $\mathscr{H}_a$. Für die Nebenbedingungen, die einen Überholvorgang vebieten, definieren wir die Menge $\mathscr{H}_o$. Damit ergibt sich auch die Menge aller relaxierten Nebenbedingungen.

$$\mathscr{H} := \mathscr{H}_d \cup \mathscr{H}_a \cup \mathscr{H}_o$$

Notation 5.3. *Sei $h \in \mathscr{H}$. Wir nennen $\lambda_h \geq 0$ den **Lagrangemultiplikator** der Nebenbedingung h. Er stellt bei Nichterfüllung der Nebenbedingung h, dessen Bestrafung dar.*

Wenn wir uns eine Nebenbedingung h aus $\mathscr{H}_d$ nehmen, sieht der „Bestrafungsterm“ wie folgt aus.

$$\lambda_h \cdot \left(1 - \sum_{w \in N_{d_i}(u)} y_w\right) \text{ mit } i \in S\backslash\{1\} \text{ und } u \in U^i$$

Dies ergibt sich recht ähnlich bei Nebenbedingungen $h \in \mathscr{H}_a$.

$$\lambda_h \cdot \left(1 - \sum_{w \in M_{a_i}(u)} y_w\right) \text{ mit } i \in S\backslash\{s\} \text{ und } u \in W^i$$

Fasst man diese beiden Typen nun zusammen, ergibt sich als erster „Bestrafungsblock“ folgendes.

$$\begin{aligned}
&\sum_{i \in S\backslash\{1\}} \sum_{u \in U^i} \lambda_h \cdot \left(1 - \sum_{w \in N_{d_i}(u)} y_w\right) + \sum_{i \in S\backslash\{s\}} \sum_{u \in W^i} \lambda_h \cdot \left(1 - \sum_{w \in M_{a_i}(u)} y_w\right) \\
= &\sum_{i \in S\backslash\{1\}} \sum_{u \in U^i} \left[\lambda_h - \lambda_h \cdot \sum_{w \in N_{d_i}(u)} y_w\right] + \sum_{i \in S\backslash\{s\}} \sum_{u \in W^i} \left[\lambda_h - \lambda_h \cdot \sum_{w \in M_{a_i}(u)} y_w\right] \\
= &\sum_{h \in \mathscr{H}_a \cup \mathscr{H}_d} \lambda_h - \sum_{i \in S\backslash\{1\}} \sum_{u \in U^i} \sum_{w \in N_{d_i}(u)} \lambda_h \cdot y_w - \sum_{i \in S\backslash\{s\}} \sum_{u \in W^i} \sum_{w \in M_{a_i}(u)} \lambda_h \cdot y_w \\
= &\sum_{h \in \mathscr{H}_a \cup \mathscr{H}_d} \lambda_h - \underbrace{\left[\sum_{i \in S\backslash\{1\}} \sum_{u \in U^i} \sum_{w \in N_{d_i}(u)} \lambda_h \cdot y_w + \sum_{i \in S\backslash\{s\}} \sum_{u \in W^i} \sum_{w \in M_{a_i}(u)} \lambda_h \cdot y_w\right]}_{=:A}
\end{aligned}$$

Relaxieren wir jedoch eine Nebenbedingung h aus $\mathscr{H}_o$, ergibt sich der folgende „Bestrafungsterm“.

$$\lambda_h \cdot \left[3 - \left(\sum_{w \in De_j^i(v)} z_{jw} + \sum_{w \in Al_j^{i+1}(u)} z_{jw} + \sum_{w \in Dl_k^i(v)} z_{kw} + \sum_{w \in Ae_k^{i+1}(u)} z_{kw}\right)\right]$$

mit $i \in S\backslash\{s\}, v \in W^i, u \in U^{i+1}, j, k \in T$ wobei $j \neq k$. Somit ergibt sich auch der zweite „Bestrafungsblock“.

$$\sum_{j\in T}\sum_{k\in T,k\neq j}\sum_{i\in S\setminus\{s\}}\sum_{v\in W^i}\sum_{u\in U^{i+1}}\lambda_h\cdot\left[3-\left(\sum_{w\in De_j^i(v)}z_{jw}+\sum_{w\in Al_j^{i+1}(u)}z_{jw}+\sum_{w\in Dl_k^i(v)}z_{kw}+\sum_{w\in Ae_k^{i+1}(u)}z_{kw}\right)\right]$$

$$=\sum_{h\in\mathscr{H}_o}3\cdot\lambda_h-\underbrace{\sum_{j\in T}\sum_{k\in T,k\neq j}\sum_{i\in S\setminus\{s\}}\sum_{v\in W^i}\sum_{u\in U^{i+1}}\lambda_h\cdot\left(\sum_{w\in De_j^i(v)}z_{jw}+\sum_{w\in Al_j^{i+1}(u)}z_{jw}+\sum_{w\in Dl_k^i(v)}z_{kw}+\sum_{w\in Ae_k^{i+1}(u)}z_{kw}\right)}_{=:B}$$

Mit Hilfe dieser beider Bestrafungsblöcke lässt sich nun die Zielfunktion unseres relaxierten LP's angeben.

$$\max\sum_{j\in T}\sum_{a\in A^j}p_a\cdot x_a+\sum_{h\in\mathscr{H}}b_h\cdot\lambda_h-A-B$$

mit

$$b_h:=\begin{cases}1; & h\in\mathscr{H}_a\cup\mathscr{H}_d\\ 3; & h\in\mathscr{H}_o\end{cases}$$

Die Unübersichtlichkeit der neu entstandenen Zielfunktion motiviert nun eine nähere Betrachtung von A und B. Wir beginnen mit A und stellen fest, dass wir den Knoten w immer dann mit λ_h bestrafen, wenn dieser in einer Umgebung $N_{d_i}(u)$ oder $M_{a_i}(u)$ auftritt. Die Idee wäre, diese ganzen Bestrafungen zusammenzufassen und somit für jeden Knoten eine Bestrafung zu erhalten. Dazu suchen wir zunächst die Nebenbedingungen, die einen Knoten betreffen.

$$H_{ad}^{(u)}:=\{h\in\mathscr{H}_a:u\in M_{a_i}(w)\}\cup\{h\in\mathscr{H}_d:u\in N_{d_i}(w)\}\text{ mit }u\in V$$

Dabei ist zu beachten, dass a_i, d_i und w durch die jeweilige Nebenbedingung h bestimmt sind.

Wir können nun die Bestrafung $\overline{q}_v$ eines Knotens v mit Hilfe der λ_h's angeben.

$$\overline{q}_v=\sum_{h\in H_{ad}^{(v)}}\lambda_h\text{ mit }v\in V$$

Damit lässt sich die Menge A recht übersichtlich darstellen, da man die Bestrafung eines Knoten gesammelt hat.

$$\begin{aligned}A&=\sum_{i\in S\setminus\{1\}}\sum_{u\in U^i}\sum_{w\in N_{d_i}(u)}\lambda_h\cdot y_w+\sum_{i\in S\setminus\{s\}}\sum_{u\in W^i}\sum_{w\in M_{a_i}(u)}\lambda_h\cdot y_w\\&=\sum_{i\in S\setminus\{1\}}\sum_{u\in U^i}\overline{q}_u\cdot y_u+\sum_{i\in S\setminus\{s\}}\sum_{u\in W^i}\overline{q}_u\cdot y_u\\&=\sum_{v\in V\setminus\{\sigma,\tau\}}\overline{q}_v\cdot y_v\end{aligned}$$

Wir gehen genauso für die Menge B vor, da diese ebenfalls zu unübersichtlich ist. Wir sammeln wiederum die Nebenbedingungen, die durch ein Zug-Knoten-Paar betroffen sind, um mit diesen anschließend die Bestrafungen für ein solches Paar angeben zu können.

$$H_o^{(j,u)}:=\{h\in\mathscr{H}_o:u\in De_j^i(v)\cup Al_j^{i+1}(w)\}\cup\{h\in\mathscr{H}_o:u\in Dl_j^i(v)\cup Ae_j^{i+1}(w)\}\text{ mit }j\in T,u\in A^j$$

wobei auch hier i, v und w durch die jeweilige Nebenbedingung h bestimmt sind. Wir geben nun den Bestrafungsfaktor an.

$$\overline{r}_{jv}:=\sum_{h\in H_o^{(j,v)}}\lambda_h\text{ mit }j\in T,v\in A^j$$

Nun lässt sich auch die Menge B mit Hilfe von $\overline{r}_{jv}$ recht übersichtlich darstellen.

$$B=\sum_{j\in T}\sum_{v\in V\setminus\{\sigma,\tau\}}\overline{r}_{jv}\cdot z_{jv}$$

Nachdem wir die relaxierte Zielfunktion entwickelt haben, können wir das relaxierte LP angeben.

Lineares Programm 5.4.

$$\max \sum_{j \in T} \sum_{a \in A^j} p_a \cdot x_a - \sum_{v \in V \setminus \{\sigma,\tau\}} \overline{q}_v \cdot y_v - \sum_{j \in T} \sum_{v \in V \setminus \{\sigma,\tau\}} \overline{r}_{jv} \cdot z_{jv} + \sum_{h \in \mathscr{H}} b_h \cdot \lambda_h \tag{19}$$

s.t.:

Flussbeschreibung:

$$\sum_{a \in \delta_j^+(\sigma)} x_a \leq 1 \tag{20}$$

für alle $j \in T$

$$\sum_{a \in \delta_j^-(v)} x_a = \sum_{a \in \delta_j^+(v)} x_a \tag{21}$$

für alle $j \in T,\ v \in V \setminus \{\sigma, \tau\}$

Variablendefinitionen:

$$z_{jv} = \sum_{a \in \delta_j^-(v)} x_a \tag{22}$$

für alle $j \in T,\ v \in V^j$

$$y_v = \sum_{j \in I_v} z_{jv} \tag{23}$$

für alle $v \in V$

$$x_a \geq 0,\ x_a \in \mathbb{Z} \tag{24}$$

für alle $a \in A$

Wir haben unser LP mit Hilfe einer Lagrange-Relaxation auf ein Multi-Commodity-Flow Problem [4] zurückgeführt. Da sich unsere Zielfunktion verändert hat, müssen wir einen Lagrangenutzen eines Pfades bzgl. eines Zuges angeben.

Definition 5.4. *Sei* $j \in T$ *und* $P^j \in \mathscr{P}^j(G)$*. Dann nennen wir* $N^L : \mathscr{P}^j(G) \to \mathbb{R}$ *den* ***Lagrangenutzen*** *des Pfades* P^j*.*

$$N^L(P^j) := N(P^j) - \sum_{v \in K(P^j)} \overline{q}_v \cdot z_{jv} - \sum_{v \in K(P^j)} \overline{r}_{jv} \cdot z_{jv}$$

Im nächsten Kapitel können wir aufbauend auf dieses relaxierte LP und den Lagrangenutzen einen Lösungsalgorithmus entwickeln, der uns eine heuristische Lösung liefert. Der Vorteil zu dem vorherigen Modell ist, dass wir die Bestrafung für Knoten bzw. Knoten-Zug-Paare haben, was sich deutlich einfacher behandeln lässt. Wir haben $|V|+|V|\cdot|T|$ Bestrafungen, was einer Größenordnung von $O(|V| \cdot |T|)$ entspricht.

6 Lösungsalgorithmus

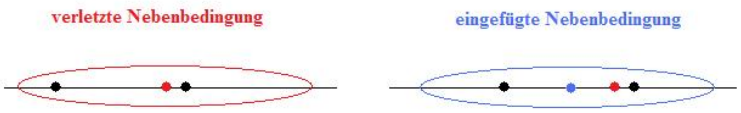
Die Idee ist möglichst gute Lagrangemultiplikatoren zu bestimmen, mit deren Hilfe wir obere Schranken für das Lineare Programm 5.3 erhalten. Zusätzlich werden wir durch eine Heuristik Lösungen erhalten, die wiederum eine untere Schranke liefern. Auch hierfür nutzen wir den definierten Lagrangenutzen. Das Ziel ist dann, mit den heuristischen Lösungen möglichst nahe an die obere Schranke heran zu kommen und die obere Schranke möglichst klein zu bekommen. Die Lagrangemultiplikatoren erhalten wir durch ein abgewandeltes Subgradientenverfahren [5].

6.1 Subgradientenverfahren

In diesem Kapitel entwickeln wir den eigentlichen Lösungsalgorithmus unseres TTP. Er basiert auf dem relaxierten Programm 5.4. Wir bestimmen zunächst eine Lösung des LP's mit den aktuellen Lagrangemultiplikatoren und betrachten anschließend die verletzten relaxierten Nebenbedingungen.Wir führen ein Update der Lagrangemultiplikatoren genau für diese Nebenbedingungen durch. Durch die relaxierte Lösung ergeben sich obere Schranken für das LP 5.3, die immer besser werden. Sollte eine Verbesserung für eine bestimmte Anzahl an Iterationen ausbleiben, wird die Schrittweite ρ im Subgradientenverfahren halbiert. Unterschreitet die Schrittweite einen Schwellenwert wird ein Fixing/Removing-Algorithmus gestartet, den wir in den Folgekapiteln erklären. Für das Update der Lagrangemultiplikatoren benutzen wir eine bekannte Formel von Held und Karp [6]

$$\lambda_h = \max\{\lambda_h + \rho \cdot \frac{UB - LB}{||g||_2^2} \cdot g_h, 0\}$$

wobei ρ die Schrittweite wie oben ist. Wir nennen unseren jeweiligen Subgradientenvektor g mit den Einträgen g_h für die jeweilige Nebenbedingung. Durch eine heuristische Vorgehensweise erhalten wir Lösungen, die uns untere Schranken liefern. Auch dieser heuristische Algorithmus wird in den Folgekapiteln vorgestellt.
Die Idee ist, drei Mengen (für Abfahrt, Ankunft und Überholen) zu initialisieren, in denen wir bereits (in vorherigen Iterationen) betrachtete Nebenbedingungen speichern. Man schaut sich die Knoten an, die eine zu enge Ab- bzw. Einfahrt in einen Bahnhof haben und prüft, ob sie in der Umgebung einer bereits betrachteten Nebenbedingung liegen. Ist dies der Fall, erfolgt ein Update dieses Lagrangemultiplikators. Ist dies nicht der Fall, fügt man eine möglichst gute Nebenbedingung in den „Pool" der betrachteten Nebenbedingungen hinzu. Dazu wählt man den Knoten, der möglichst gleichweit entfernt ist von den beiden betroffenen Knoten und fügt die dazugehörige Nebenbedingung dem „Pool" hinzu.



Eine ähnliche Vorgehensweise wählen wir auch bei den verbotenen Überholvorgängen. Überholen sich zwei Züge auf einer Strecke, wird zunächst geschaut, ob der „Pool" bereits eine Nebenbedingung enthält, die diese vier Knoten enthält. Ist dies der Fall, wird deren Lagrangemultiplikator aktualisiert. Ansonsten fügen wir eine Nebenbedingung dem Pool hinzu. Dazu wählen wir einen Knoten im Abfahrtsbahnhof, der von den beiden Abfahrtsknoten möglichst gleichweit entfernt ist und einen Knoten im Ankunftsbahnhof, der von den beiden betroffenen Ankunftsknoten möglichst gleichweit entfernt ist. Wir fügen nun die Nebenbedingung aus $\mathscr{H}_o$, die zu diesen Knoten gehört, dem „Pool" hinzu.

Algorithmus 6.1. *Das Subgradientenverfahren*

Input: *ein LP der Form 5.4, Menge der relaxierten Nebenbedingungen $\mathscr{H}$, Schwellenwert R, Halbierungszahl N, Initialschrittweite ρ_0*

Output: *eine obere Schranke für den Zielfunktionswert von 5.4 und eine heuristische Lösung*

(1) *Initialisisere $\lambda_h = 0, k = 0, \rho = \rho_0$ und $H_a = \emptyset, H_d = \emptyset, H_o = \emptyset$*
(2) *Ermittle Lösung x_0 durch Lösen von LP 5.4 mit aktuellen λ_h's. Sei ZF der Zielfunktionswert und f der zugehörige Fahrplan.*
(3) *wenn $ZF < UB$, dann setze $UB = ZF$ und $k = 0$*
(4) *wenn $ZF \geq UB$, dann setze $k = k + 1$*
(5) *wenn $k = N$, dann setze $\rho = \frac{\rho}{2}$ und $k = 0$*

(6) *wenn $\rho < R$, dann starte* ***Fixing/Removing-Algorithmus***

(7) *Berechne evtl. neue untere Schranke(LB) mit dem* ***heuristischen Lösungsalgorithmus*** *mit aktuellen λ_h's*

(8) *Für alle u_1, u_2, die in A_f besucht werden, mit $u_1, u_2 \in U^i$ ($i \in S\backslash\{1\}$) und $\Delta(u_1, u_2) < a_i$* ***(Verletzung der Ankunft)***

(a) *wenn ein $h_0 \in H_a$ existiert mit $u \in U^i$ und $u_1, u_2 \in M_{a_i}(u)$*

$$\lambda_{h_0} = \max\{\lambda_{h_0} + \rho \cdot \frac{UB - LB}{||g||_2^2} \cdot g_{h_0}, 0\}$$

(b) *wenn nicht, füge $h_0 \in \mathscr{H}_a$ mit $M_{a_i}(u)$ in H_a ein, wobei u so gewählt ist, dass*

$$u \in \arg \min_{u_1 \preceq u' \preceq u_2} |\Delta(u_1, u') - \Delta(u', u_2)|$$

und

$$\lambda_{h_0} = \max\{\lambda_{h_0} + \rho \cdot \frac{UB - LB}{||g||_2^2} \cdot g_{h_0}, 0\}$$

(9) *Für alle u_1, u_2, die in A_f besucht werden, mit $u_1, u_2 \in W^i$ ($i \in S\backslash\{s\}$) und $\Delta(u_1, u_2) < d_i$* ***(Verletzung der Abfahrt)***

(a) *wenn ein $h_0 \in H_d$ existiert mit $u \in W^i$ und $u_1, u_2 \in N_{d_i}(u)$*

$$\lambda_{h_0} = \max\{\lambda_{h_0} + \rho \cdot \frac{UB - LB}{||g||_2^2} \cdot g_{h_0}, 0\}$$

(b) *wenn nicht, füge $h_0 \in \mathscr{H}_d$ mit $N_{d_i}(u)$ in H_d ein, wobei u so gewählt ist, dass*

$$u \in \arg \min_{u_1 \preceq u' \preceq u_2} |\Delta(u_1, u') - \Delta(u', u_2)|$$

und

$$\lambda_{h_0} = \max\{\lambda_{h_0} + \rho \cdot \frac{UB - LB}{||g||_2^2} \cdot g_{h_0}, 0\}$$

(10) *Für alle $(v_1, u_1), (v_2, u_2) \in A_f$ mit $v_1, v_2 \in W^i (i \in S\backslash\{s\}), u_1, u_2 \in U^{i+1}, (v_1, u_1) \in A^j (j \in T), (v_2, u_2) \in A^k (k \in T)$ und $v_1 \preceq v_2, u_2 \preceq u_1$* ***(Verletzung des Überholverbots)***

(a) *wenn ein $h_0 \in H_o$ existiert mit $v \in W^i, u \in U^{i+1}$ und $v_1 \in De_j^i(v), v_2 \in Dl_k^i(v), u_1 \in Al_j^{i+1}(u), u_2 \in Ae_k^{i+1}(u)$*

$$\lambda_{h_0} = \max\{\lambda_{h_0} + \rho \cdot \frac{UB - LB}{||g||_2^2} \cdot g_{h_0}, 0\}$$

(b) *wenn nicht, füge $h_0 \in \mathscr{H}_o$ mit $v \in W^i, u \in U^{i+1}$ ein, wobei u und v so gewählt sind, dass*

$$u \in \arg \min_{u_2 \preceq u' \preceq u_1} |\Delta(u_2, u') - \Delta(u', u_1)| \text{ und } v \in \arg \min_{v_1 \preceq v' \preceq v_2} |\Delta(v_1, v') - \Delta(v', v_2)|$$

und

$$\lambda_{h_0} = \max\{\lambda_{h_0} + \rho \cdot \frac{UB - LB}{||g||_2^2} \cdot g_{h_0}, 0\}$$

(11) *Gehe zu (2)*

In den nächsten Kapiteln entwickeln wir nun die zwei noch fehlenden Algorithmen, um den oben vorgestellten Algorithmus zu komplettieren.

6.2 Heuristischer Lösungsalgorithmus

Dieses Kapitel besteht aus einem Algorithmus der uns aus einer Lösung des relaxierten Problems 5.4 eine heuristische Lösung für das LP 5.3 liefert. Dabei werden die Züge absteigend nach dem Lagrangenutzen der Pfade, mit denen sie in der relaxierten Lösung geplant wurden, sortiert. Es wird nun von oben jeder Zug mit dem lagrangenutzenmaximalen Pfad geplant und anschließend in dem Graph die zu diesem Pfad inkompatiblen Kanten entfernt. In diesem neuen Graph wird nun für den nächsten Zug ein lagrangenutzenmaximale Pfad gesucht. Dies wird solange wiederholt, bis jeder Zug betrachtet wurde. Wir nutzen den Lagrangenutzen, da dadurch Züge, die in der Reihenfolge weiter hinten sind, auch mit berücksichtigt werden. Wird kein Pfad gefunden oder hat der gefundene Pfad einen negativen Nutzen (nicht Lagrangenutzen), wird der Zug nicht geplant.
Um die Vorgehensweise zu beschleunigen, betrachten wir nur eine Teilmenge der Züge. Wir unterteilen die Züge in drei Gruppen. Die erste Gruppe sind die **fixierten** Züge, welche in der Lösung einen vorgegebenen Pfad haben, mit denen sie geplant werden und welcher auch nicht geändert wird. Die zweite Gruppe sind die **entfernten** Züge, welche in der Lösung nicht betrachtet werden, also keinen Pfad zugewiesen bekommen. Und schließlich die Gruppe der **freien** Züge, die wir so behandeln, wie es oben beschrieben wurde. Wie sich diese drei Gruppen zusammensetzen, werden wir im Zuge des Kapitels 6.3 erläutern. Am Anfang sind alle Züge als frei zu sehen. Der Pfad, der einem fixierten Zug j zugewiesen wird, nennen wir P^j_{fix}.
Dieser Algorithmus ist im Folgenden aufgeschrieben. Nach diesem Algorithmus wird ein Verfeinerungsalgorithmus gestartet. Diesen erklären wir anschließend.
Wie oben erwähnt, werden im Verlauf des Algorithmus inkompatible Kanten entfernt. Dies motiviert die Definition einer Menge, die alle Kanten einer Kantenmenge $A_1 \subseteq A$ enthält, die zu einer Kantenmenge $A_2 \subseteq A$ inkompatibel sind.

$$Ink(A_1, A_2) := \{a \in A_1 : \exists a_0 \in A_2 : a \text{ und } a_0 \text{ sind inkompatibel}\}$$

Algorithmus 6.2. *Basisalgorithmus*

Input: *aktuelle Lösung $f \in \mathscr{F}$ von LP 5.4, Graph G=(A,V), Züge T*

Output: *heuristische Lösung von LP 5.3*

(1) *Initialisiere $T^{akt} = T, A^{akt} = A, G^{akt} = (A^{akt}, V), F = \emptyset$, Fahrplan $f_0 \in \mathscr{F}$ mit $A_{f_0} = \emptyset$*
(2) *Plane alle fixierten Züge j mit dem Pfad P^j_{fix}, also setze $A_{f_0} = \bigcup_{j\ fix} P^j_{fix}$.*
Setze $A^{akt} = A \setminus \bigcup_{j\ fix} Ink(A, P^j_{fix})$ und $G^{akt} = (A^{akt}, V)$
(3) *Plane alle entfernten Züge nicht.*
(4) *Setze $T^{akt} = T \setminus \left(\bigcup_{j\ fix} j \cup \bigcup_{j\ entf.} j \right)$.*
(5) *Wenn $T^{akt} = \emptyset$, dann **Verfeinerungsalgorithmus** mit f_0 und Menge der geplanten Züge F*
(6) *Wähle ein $j^* \in \arg\max_{j \in T^{akt}} N^L(P^j_f)$*
(7) *Wenn $\mathscr{P}^{j^*}(G^{akt}) = \emptyset$, dann wird Zug j^* nicht geplant. Gehe zu (11).*
(8) *Wähle lagrangenutzenmaximalen Pfad $P_0^{j^*} \in \arg\max_{p \in \mathscr{P}^{j^*}(G^{akt})} N^L(p)$ zu Zug j^**
(9) *Wenn $N(P_0^{j^*}) \leq 0$, wird der Zug nicht geplant. Gehe zu (11).*
(10) *Zug j^* wird mit Pfad $P_0^{j^*}$ geplant, also setze $A_{f_0} = A_{f_0} \cup P_0^{j^*}$.*
Setze $F = F \cup \{j^*\}, A^{akt} = A^{akt} \setminus Ink(A^{akt}, P_0^{j^*})$ und $G^{akt} = (A^{akt}, V)$
(11) *Setze $T^{akt} = T^{akt} \setminus \{j^*\}$. Gehe zu (5).*

Wenn der Basisalgorithmus 6.2 durchgelaufen ist, versuchen wir die erhaltene Lösung noch etwas zu verbessern. Dazu betrachten wir zunächst alle geplanten freien Züge. Wir suchen für jeden von diesen den nutzenmaximalen (nicht Lagrangenutzen !!!) Pfad, der kompatibel zu den bereits geplanten Zügen ist und planen den Zug mit diesem Pfad in dem Fahrplan ein. Wir können hier auf eine Positivitätsbetrachtung des Nutzens verzichten, da diese dadurch gesichert ist, dass der Zug bereits geplant wurde und somit auf jedenfall ein Pfad mit positivem Nutzen existiert.
Anschließend betrachten wir die freien Züge, die noch nicht eingeplant wurden. Auch bei diesen suchen wir nach nutzenmaximalen Pfaden, die kompatibel zu den bereits geplanten Zügen sind, und planen sie, falls der Nutzen positiv ist, ein.

Dies kann passieren, da möglicherweise durch Veränderungen bei den bereits geplanten Zügen nun nutzenpositive Pfade auftreten.

Algorithmus 6.3. *Verfeinerungsalgorithmus*

Input: *Lösung f des Basisalgorithmus, Graph $G = (A, V)$, Menge der geplanten Züge F*

Output: *heuristische Lösung von LP 5.3*

(1) *Initialisiere* $F^{akt} = F$

(2) *Wenn* $F^{akt} = \emptyset$, *dann gehe zu (6)*

(3) *Wähle* $j^* \in F^{akt}$ *und setze* $A^{akt} = A \setminus \bigcup_{j \in T \setminus \{j^*\}} Ink(A, P_f^j)$, $G^{akt} = (A^{akt}, V)$ *sowie* $A_f = A_f \setminus P_f^{j^*}$

(4) *Wähle nutzenmaximalen Pfad* $P_0^{j^*} \in \arg\max_{p \in \mathscr{P}^{j^*}(G^{akt})} N(p)$ *zu Zug* j^* *und setze* $A_f = A_f \cup P_0^{j^*}$

(5) *Setze* $F^{akt} = F^{akt} \setminus \{j^*\}$. *Gehe zu (2).*

(6) *Initialisiere* $N^{akt} = \bigcup_{j \text{ frei}} j \setminus F$.

(7) *Setze* $A^{akt} = A \setminus \bigcup_{j \in T} Ink(A, P_f^j)$ *und* $G^{akt} = (A^{akt}, V)$

(8) *Wenn* $N^{akt} = \emptyset$, *dann STOP (Lösungsfahrplan: f mit aktiven Kanten A_f)*

(9) *Wähle* $j^* \in N^{akt}$.

(10) *Wähle nutzenmaximalen Pfad* $P_0^{j^*} \in \arg\max_{p \in \mathscr{P}^{j^*}(G^{akt})} N(p)$ *zu Zug* j^*.

(11) *Wenn* $N(P_0^{j^*}) > 0$, *wird der Zug geplant. Setze* $F = F \cup \{j^*\}$ *und* $A_f = A_f \cup P_0^{j^*}$.
Setze $A^{akt} = A^{akt} \setminus Ink(A^{akt}, P_0^{j^*})$ *und* $G^{akt} = (A^{akt}, V)$.

(12) *Setze* $N^{akt} = N^{akt} \setminus \{j^*\}$. *Gehe zu (8).*

Wir starten diesen Verfeinerungsalgorithmus iterativ immer wieder bis keine Fahrplanverbesserung mehr auftritt. Dies ist dann unsere heuristische Lösung, die wir in jedem Iterationsschritt des Subgradientenverfahrens berechnen. Der Nutzen des so entstandenen Fahrplans liefert eine untere Schranke des TTP. Wir speichern immer den Fahrplan der besten unteren Schranke als aktuelle beste Lösung.

6.3 Fixing/Removing-Algorithmus

Wie wir bereits vorher erwähnt haben, wird dieser Algorithmus gestartet, wenn die Schrittweite unter einen bestimmten Schwellenwert sinkt. Die Idee ist nun in so einer Situation bestimmte Züge zu „fixieren“, d.h. ihren Pfad bei der Lösungsfindung festzuhalten. Dadurch wird die Lösungssuche um einiges einfacher. Desweiteren werden bestimmte Züge „entfernt“, was wiederum bedeutet, dass diese Züge ausfallen und diese bei der Lösungssuche auch nicht beachtet werden müssen. Die dritte Kategorie in der ein Zug liegen kann sind die „freien“ Züge, für die wie gewöhnlich ein Pfad gesucht wird. Jedesmal, wenn ein Zug fixiert oder entfernt wird, werden zusätzlich die Lagrangemultiplikatoren abgeändert.(Diese Idee entstammt dem Paper [7].)
Der Algorithmus geht nun wie folgt vor. Wenn es freie Züge gibt, die nicht geplant wurden, werden diese entfernt. Ist dies nicht der Fall, werden die $\lceil \frac{t}{20} \rceil$ Lagrangenutzenmaximalen freien Züge fixiert, falls ihr Nutzen nichtnegativ ist.

Algorithmus 6.4. *Fixing/Removing-Algorithmus*

Input: *Die bisher beste Lösung f mit zugehörigen Lagrangemulitplikatoren λ_h, Inititialschrittweite ρ_0'*

(1) *Wenn r>0 freie Züge existieren, die in f nicht geplant sind, entferne* $\min\{r, \lceil \frac{t}{20} \rceil\}$ *der Züge mit niedrigstem Lagrangenutzen.* ***STOP****. Starte Subgradientenalgorithmus mit* $\rho = \rho_0'$ *und aktuellen Lagrangemultiplikatoren.*

(2) *Initialisiere* $k = 0, A^{akt} = A \setminus \bigcup_{j \text{ fix}} Ink(A, P_{fix}^j)$ *und* $G^{akt} = (A^{akt}, V)$

(3) *Wenn* $k = \lceil \frac{t}{20} \rceil$, *dann* ***STOP****. Starte Subgradientenalgorithmus mit* $\rho = \rho_0'$ *und aktuellen Lagrangemultiplikatoren.*

(4) *Wähle* $j^* \in \arg\max_{j \text{ frei}} \left(\max_{p \in \mathscr{P}^j(G^{akt})} N^L(p) \right)$ *mit* $P_0^{j^*} \in \arg\max_{p \in \mathscr{P}^j(G^{akt})} N^L(p)$.

(5) *Wenn* $N(P_0^{j^*}) > 0$*, wird* j^* *mit* $P_0^{j^*}$ *fixiert und setze* $k = k + 1$.

(6) *Setze* $A^{akt} = A^{akt} \backslash Ink(A^{akt}, P_{fix}^{j^*})$ *und* $G^{akt} = (A^{akt}, V)$.

(7) *Gehe zu (3).*

Wir führen nun bei jedem Fixieren und Entfernen ein Update der Lagrangemultiplikatoren durch. Wird also der Zug j fixiert oder entfernt, werden alle Lagrangemultiplikatoren zu Nebenbedingungen aus $\mathscr{H}_o$, die den Zug j beinhalten auf 0 gesetzt.
Auch für Nebenbedingungen aus $\mathscr{H}_a$ und $\mathscr{H}_d$ werden die Multiplikatoren geändert. Dazu nehmen wir uns eine beliebige Nebenbedingung und zählen für jeden freien Zug k, wie oft sein Pfad in allen Subgradienteniterationen einen Knoten in der entsprechenden Nebenbedingung besucht. Diese Zahl nennen wir Ψ_k. Wir definieren uns nun eine Menge M.

$$M := \{k \in T : \Psi_k > 0\}$$

Wenn $j \in M$ gilt und $|M| = 2$ gilt, setzen wir den Lagrangemultiplikator der betrachteten Nebenbedingung auf 0. Wenn andernfalls $j \in M$ gilt, setzen wir den entsprechenden Lagrangemultiplikator wie folgt.

$$\lambda_h = \lambda_h \cdot \left(1 - \frac{\Psi_j}{\sum\limits_{k \in M} \Psi_k}\right)$$

Falls $j \notin M$ gilt, bleibt der Langrangenutzen unverändert, da er die Nebenbedingung nicht verletzt. Dadurch wird heuristisch berücksichtigt, wie sich der Zug zu der betrachteten Nebenbedingung verhält.

Es könnte jetzt jedoch vorkommen, dass wir Züge zu unrecht fixiert haben. Diesen Fehler müssen wir korrigieren können. Dazu implementieren wir einen **unfixing-Algorithmus**. Dieser setzt die $\lceil \frac{t}{20} \rceil$ fixierten Züge mit dem niedrigsten Lagrangenutzen auf frei, sowie die $\lceil \frac{t}{20} \rceil$ entfernten Züge mit dem höchsten Lagrangenutzen auf frei.
Jetzt ist noch die Frage, wie man merkt, dass man in die falsche Richtung läuft. Dazu berechnen wir mit Hilfe des relaxierten LP's 5.4 verbunden mit den fixierten bzw. entfernten Zügen eine obere Schranke für das LP 5.3 verbunden mit den fixierten bzw. entfernten Zügen. Diese obere Schranke gibt uns einen maximalen Wert, den wir in unserem aktuellen Zustand maximal erreichen können. Ist diese obere Schranke jedoch kleiner als der Nutzen, der aktuell besten gefundenen Lösung, macht es keinen Sinn in diese Richtung weiter zu suchen und wir starten den oben beschriebenen unfixing-Algorithmus. Nach diesem unfixing-Algorithmus starten wir den Subgradientenalgorithmus mit ρ_0' und den aktuell besten Lagrangemultiplikatoren.
Taucht nun zwischen dem Aufruf des unfixing-Algorithmus und dem erneuten Unterschreiten des Schrittweitenschwellenwertes keine neue untere Schranke (verbesserte Lösung) auf, werden alle Züge auf frei gesetzt und der Subgradientenalgorithmus mit den aktuell besten Lagrangemultiplikatoren und ρ_0' gestartet.
Der Algorithmus terminiert, wenn eine gewünschte Zeit oder eine gewünschte Anzahl an Iterationen durchgelaufen ist.

Damit ist es nun gelungen eine heuristische Lösung zu finden. Dadurch dass diese durch obere Schranken „abgesichert“ ist, kann man recht gut die Güte dieser Lösung erkennen, was uns eine Sicherheit gibt, wie stark wir uns der Optimallösung genähert haben. Praktische Ergebnisse dieses Verfahrens lassen sich dem Paper [1] entnehmen.
Es muss bei der Lösung jedoch berücksichtigt werden, dass wir Vereinfachungen vorrausgesetzt haben. Wir haben lediglich eine isolierte Strecke betrachtet, was in der Realität durchaus problematisch werden könnte. Auch haben wir eine recht einfache Wahl des Nutzens getroffen, da dieser lediglich von Stretch und Shift abhängt. Jedoch haben wir mit diesen Einschränkungen eine akzeptable Lösung gefunden.

Literatur

[1] Alberto Caprara, Matteo Fischetti, and Paolo Toth. Modeling and solving the train timetabling problem. *Operations Research 50*, pages 851–861, 2002.

[2] Th. H. Corman, Ch. E. Leiserson, R. Rivest, and C. Stein. *Algorithmen - Eine Einführung.* Oldenbourg, 2001.

[3] M.R. Garey and D.S. Johnson. *Computers and Intractability: A Guid to the Theory of NP-Completeness.* W.H. Freeman, San Francisco, 2003.

[4] Ravindra K. Ahuja, Thomas L. Magnati, and James B. Orlin. *Network Flows: Theory, Algorithms, and Applications.* Prentice Hall, 1993.

[5] A. Martin and S. O. Krumke. *Diskrete Optimierung.* Springer, 2009. noch nicht erschienen.

[6] M. Held and R.M. Karp. The traveling salesman problem and minimum spanning trees: Part ii. *Mathematical Programming 1*, pages 6–25, 1971.

[7] A. Caprara, M. Fischetti, and P. Toth. A heuristic method for the set covering problem. *Operations Research 47*, pages 730–743, 1999.