Technical Report 1-2017

Vereinigung von Intervallmengen für minimale Anzahl von Mindestzugfolgezeiten zu fixierten Trassen

Peter Großmann, Reyk Weiß

7. November 2017

1 Einleitung

In der automatischen Konstruktion von Schienengüterverkehrs-Systemtrassen [6, 1] ist einer der Haupteinflussfaktoren die kapazitative Einschränkung von vorkonstruierten, fixierten Trassen (Personenverkehr und Güterverkehr). Dieser spiegelt sich in Mindestzugfolgezeiten [4] zu den neu zu konstruierenden Trassen wider. Diese sind jeweils als zwei zusätzliche, lineare Restriktionen im linearen Modell enthalten. [2]

Ziel dieser Arbeit ist es, die Anzahl an Mindestzugfolgezeiten von neu zu konstruierenden Systemtrassen zu vorkonstruierten Trassen zu minimieren. Betrachtet man beispielsweise Abb. 1 lässt sich schnell deduzieren, dass die Kapazität in bestimmten Bereichen so stark eingeschränkt ist, dass sich hier mehrere Abschnitte (in der Abbildung das gelbe Polygon) zu einem einzigen zusammenfassen lassen. Diese Abschnitte sind zeitlich und räumlich belegte Bereiche, in denen die neu zu konstruierenden Systemtrassen nicht liegen dürfen. Überschneiden sich diese Bereiche (Blöcke), kann man sie zu einem Block vereinigen und somit die Anzahl der Mindestzugfolgezeiten verkleinern. In Abbildung 1 lässt sich erkennen, dass die neu einzulegende Systemtrasse (bzw. der Teil selbiger) entweder vor dem Block an vorkonstruierten Trassen (gelber Bereich) oder danach geplant werden muss.

Diese zeitlichen Bereiche lassen sich mathematisch als Intervalle ausdrücken [5]. Beispielsweise wird die Zusammenfassung beziehungsweise Vereinigung von Intervallen in Abbildung 2 dargestellt. Repräsentiert jedes Intervall eine Mindestzugfolgezeit, würde in

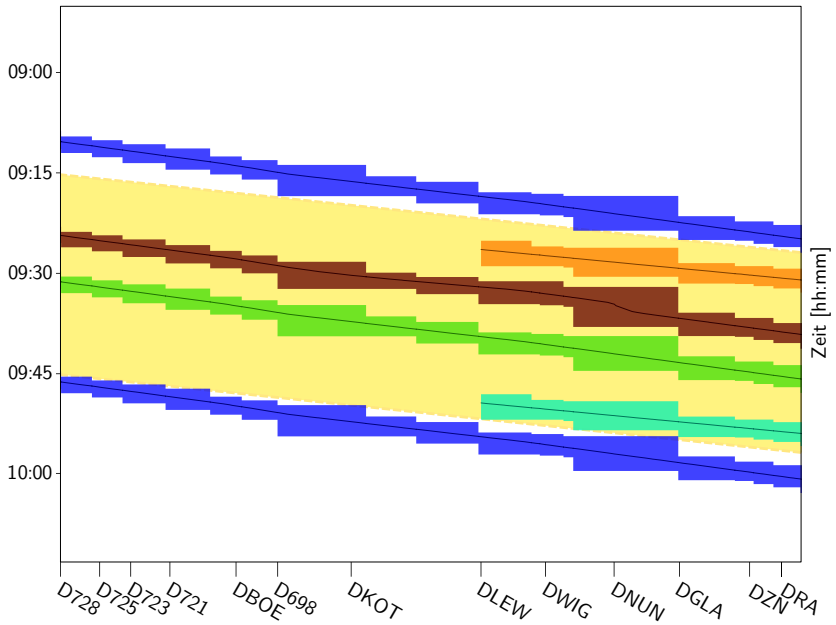

Abbildung 1: Screenshot aus TAKT mit Bildfahrplan (Sperrtreppen) mit Teil einer neu zu konstruierenden Systemtrasse (dunkelblau) zu vorkonstruiertem Verkehr.

der Abbildung somit die Anzahl von vier auf zwei verkleinert werden. Die tatsächliche Reduzierung wird anhand von realen Testszenarien aufgezeigt.

Da es sich im Rahmen dieser Veröffentlichung um eine Machbarkeitsstudie handelt, wird auf Beweise zunächst verzichtet. Obwohl Intervalle in dieser Arbeit ganzzahlig sind, werden sie in Darstellungen reell (durchgezogene Linien) angezeigt, um eine anschaulicheres Verständnis zu erhalten.

2 Vorüberlegungen und Definitionen

Bevor man ganzzahlige Intervalle geordnet (d. h. in Reihenfolge) vereinigen kann, benötigt man zunächst die formale Definition eines ganzzahligen Intervalls. Die meisten Erkenntnisse über Intervalle sind bereits in der Literatur erarbeitet worden, werden aber hier zwecks Anwendung nochmals aufgeführt [8].

Definition 1 (Ganzzahliges Intervall)**.** *Es seien $a, b \in \mathbb{Z}$ ganze Zahlen. Dann heißt*

$$[a, b] := \{c \in \mathbb{Z} \mid a \leq c \leq b\} \subset \mathbb{Z}$$

ganzzahliges Intervall.

Im Folgenden werden ganzzahlige Intervalle lediglich Intervalle genannt. Die Menge aller möglichen ganzzahligen Intervalle wird mit $\mathcal{I}$ bezeichnet.

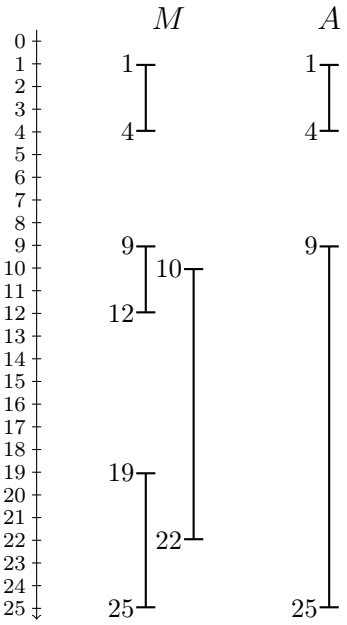

Abbildung 2: Vereinigung A der Intervallmenge $M = \{[1,4],[19,25],[9,12],[10,22]\}$.

Beispiel 2. Seien $[1,4] \in \mathcal{I}$ und $[-3,2] \in \mathcal{I}$ Intervalle. Dann gilt laut Definition 1

$$[1,4] = \{1,2,3,4\},$$
$$[-3,2] = \{-3,-2,-1,0,1,2\}.$$

Lemma 3 (Schnitt von Intervallen (ohne Beweis))**.** *Der Schnitt ($\cap$) zweier Intervalle ist wieder ein Intervall.*

Beispiel 4. Sei $[1,4]$ und $[-3,2]$ (siehe Beispiel 2) Intervalle. Dann gilt:

$$[1,4] \cap [-3,2] = \{1,2,3,4\} \cap \{-3,-2,-1,0,1,2\} = \{1,2\} = [1,2]$$

und $[1,2]$ ist ein Intervall (veranschaulicht in Abb. 3). Schneidet man die disjunkten Intervalle $[1,2]$ und $[5,6]$ ergibt sich $[1,2] \cap [5,6] = \emptyset$. Dies ist auch ein Intervall und steht nicht im Widerspruch mit Definition 1 bzw. Lemma 3.

Definition 5 (Partiell Geordnete Menge)**.** *Sei M eine Menge und $\leq$ eine Relation. Dann heißt* $(M, \leq)$ partiell geordnete Menge *[8], wenn für alle $x, y, z \in M$ gilt*

$x \leq x$	*(Reflexifität)*
$x \leq y \wedge y \leq x \Rightarrow x = y$	*(Antisymmetrie)*
$x \leq y \wedge y \leq z \Rightarrow x \leq z$	*(Transitivität)*

Wir signalisieren geordnete Mengen mittels der spitzen Klammen $\langle$ und $\rangle$.

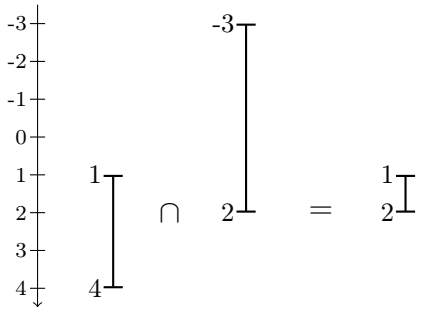

Abbildung 3: Schnitt der Intervalle $[1, 4]$ und $[-3, 2]$.

Beispiel 6. Sei $M = \{4, 1, 3, 9\}$ eine Menge von ganzen Zahlen mit der für ganzen Zahlen bekannten $\leq$-Relation. Dann ist $(M, \leq)$ eine partiell geordnete Menge mit $\langle 1, 3, 4, 9\rangle$.

Damit wir eine Menge von geordneten Intervallen nach Definition 5 benutzen können fordern wir stets an die Menge M, dass die Intervalle disjunkt sind, wenn wir sie ordnen wollen. Das heißt, wir setzen für eine Menge von Intervallen M fest, dass für alle paarweise verschiedenen Intervalle $I, J \in M$ gilt: $I \cap J = \emptyset$ (siehe Beispiel 4). Alternativ kann man M mittels disjunkter Vereinigung wie folgt definieren:

$$M = \dot{\bigcup_{I \in M}} I$$

Wir definieren für Intervalle hier die $\leq$-Relation als

$$[a, a'] \leq [b, b'] :\Leftrightarrow a \leq b \tag{1}$$

Lemma 7 (Partielle Ordnung von Intervallen (ohne Beweis))**.** *Sei M eine Menge von disjunkten Intervallen und $\leq$ die Relation aus* (1)*. Dann ist $(M, \leq)$ eine partiell geordnete Menge.*

Beispiel 8. Es sei $M = \{[1, 3], [9, 12], [5, 7]\} \subseteq \mathcal{I}$ eine Menge von disjunkten Intervallen. Dann gilt für die $\leq$-Relation aus (1) mittels Lemma 7, dass $(M, \leq)$ eine partiell geordnete Menge ist mit

$$\langle [1, 3], [5, 7], [9, 12]\rangle,$$

da $1 \leq 5 \leq 9$.

3 Vereinigung von Intervallmengen

Man kann aus jeder ganzzahligen Menge eine Menge disjunkter Intervalle erzeugen. Folgendes Beispiel soll die zunächst mehreren Lösungsmöglichkeiten aufzeigen.

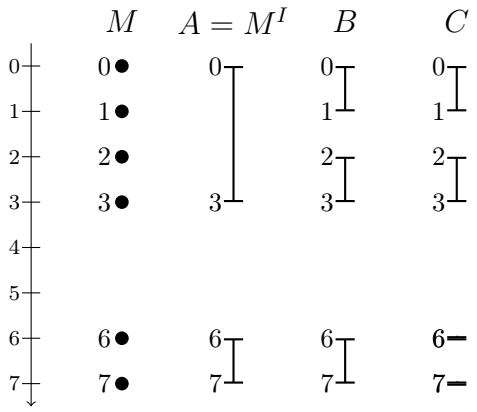

Abbildung 4: Mögliche Intervallmengen für die ganzzahlige Menge $\{0, 1, 2, 3, 6, 7\}$.

Beispiel 9. Sei $M = \{0, 1, 2, 3, 6, 7\} \subseteq \mathbb{Z}$ eine ganzzahlige Menge. Dann ergeben sich für M beispielsweise folgende Mengen ganzzahliger Intervalle

$$A = \{[0, 3], [6, 7]\},$$
$$B = \{[0, 1], [2, 3], [6, 7]\}.$$

Beide Mengen (A und B) enthalten jeweils alle Werte aus M (siehe Abb. 4).

Wir können eine Menge ganzer Zahlen aus einer Intervallmenge erhalten („flatten“) mittels folgender Definition.

Definition 10. *Sei $M \subset \mathcal{I}$ eine Intervallmenge. Dann sei M^x die Menge aller ganzzahligen Werte der Intervalle aus M, mit*

$$M^x := \bigcup_{J \in M} J.$$

Beispiel 11. Sei $A = \{[0, 3], [6, 7]\} \subseteq \mathcal{I}$ eine Intervallmenge. Dann ist $A^x = \{0, 1, 2, 3, 6, 7\}$ die Menge aller ganzzahligen Werte wie in Beispiel 9 die Menge M.

Um der Motivation, minimale Anzahl an Zugfolgezeiten, gerecht zu werden, erstellen wir nur die Intervallmengen, die eine minimale Anzahl von Intervallen enthält.

Definition 12 (Zugehörige Intervallmenge)**.** *Sei $M \subseteq \mathbb{Z}$ eine Menge ganzer Zahlen. Dann heißt M^I die* zugehörige Intervallmenge von M *mit*

$$M^I := \arg\min_{A \in \{A \subseteq \mathcal{I} \mid A^x = M\}} |A|.$$

Beispiel 13. Sei $M = \{0, 1, 2, 3, 6, 7\} \subseteq \mathbb{Z}$ die ganzzahlige Menge aus Beispiel 9. Dann ist $M^I = A = \{[0, 3], [6, 7]\}$ die zugehörige Intervallmenge von M. Weitere Mengen von

$$\{A \subseteq \mathcal{I} \mid A^x = M\}$$

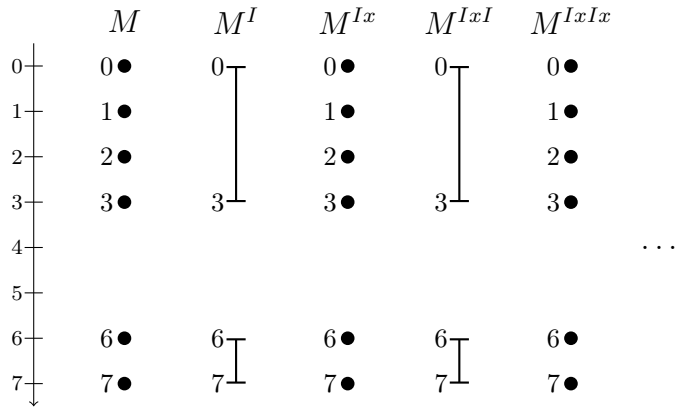

Abbildung 5: Veranschaulichung von Korollar 15 für $M = \{0, 1, 2, 3, 6, 7\}$.

wären beispielsweise

$$B = \{[0, 1], [2, 3], [6, 7]\},$$
$$C = \{[0, 1], [2, 3], [6, 6], [7, 7]\}.$$

Allerdings gilt $|A| = 2 < 3 = |B|$, $|A| = 2 < 4 = |C|$. Somit ist A die minimale Intervallmenge und deshalb auch die *zugehörige* (siehe Abb. 4).

Lemma 14 (ohne Beweis)**.** *Sei $M \subseteq \mathbb{Z}$ eine ganzzahlige Menge, dann gilt für die zugehörige Intervallmenge M^I:*

$$(i) : \forall J \in M^I \forall x \in J : x \in M$$
$$(ii) : \forall x \in M \exists J \in M^I : x \in J$$

Korollar 15 (ohne Beweis)**.** *Es folgt die Beobachtung, dass die mehrfache Anwendung der Operatoren auf eine ganzzahlige Menge M sich wieder die selben Mengen ergeben, so dass*

$$M = M^{Ix} = M^{IxIx} = \ldots,$$
$$M^I = M^{IxI} = M^{IxIxI} = \ldots.$$

Korollar 15 ist in Abb. 5 veranschaulicht.

Vereinigt man Intervalle soll hier die Definition so sein, dass stets eine partiell geordnete Menge von Intervallen entsteht.

Beispiel 16.

$$[0, 3] \cup [6, 7] = \langle [0, 3], [6, 7] \rangle$$
$$[0, 3] \cup [1, 7] = \langle [0, 7] \rangle$$

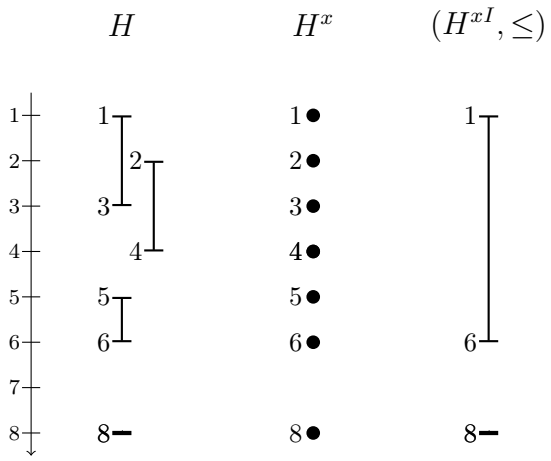

Abbildung 6: Intervallmenge H und die zugehörige partiell geordnete Intervallmenge $(H^{xI}, \leq)$.

In diesem Beispiel erkennt man wieder bereits, dass mehrere Lösungen möglich wären. Somit wollen wir auch hier die minimale Anzahl an resultierenden Intervallen erhalten. Es ergibt sich somit die finale Definition.

Definition 17 (Geordnete Vereinigung Intervalle)**.** *Es sei $M \subseteq \mathcal{I}$ eine Menge von Intervallen. Dann ist*

$$(M^{xI}, \leq)$$

die Vereinigung der Intervalle von M als partiell geordnete Menge.

Die oben genannte Forderung, dass M^{xI} disjunkt ist, ergibt sich automatisch (ohne Beweis) aus den Definitionen 10 und 12. Dies soll das folgende Beispiel veranschaulichen.

Beispiel 18. Sei $M = \{[6, 7], [0, 3]\}$ eine Intervallmenge. Dann ist $M^x = \{6, 7, 0, 1, 2, 3\}$ die ganzzahlige Menge von M und somit $M^{xI} = M = \{[6, 7], [0, 3]\}$ die zugehörige Intervallmenge. Da sich keine Intervalle überschneiden, ist dieses Beispiel noch recht uninteressant. Dennoch muss noch eine Ordnung für $(M^{xI}, \leq)$ vorgenommen werden, sodass sich final ergibt

$$(M^{xI}, \leq) = \langle [0, 3], [6, 7] \rangle.$$

Interessanter ist das folgende Beispiel. Sei $H = \{[1, 3], [5, 6], [8, 8], [2, 4]\}$ eine Intervallmenge. Dann gilt für die Vereinigung

$$\begin{aligned} & H^x = \{1, 2, 3, 5, 6, 8, 4\} \\ \Rightarrow\ & H^{xI} = \{[8, 8], [1, 6]\} \\ \Rightarrow\ & (H^{xI}, \leq) = \langle [1, 6], [8, 8] \rangle. \end{aligned}$$

Dieser Sachverhalt ist in Abb. 6 veranschaulicht.

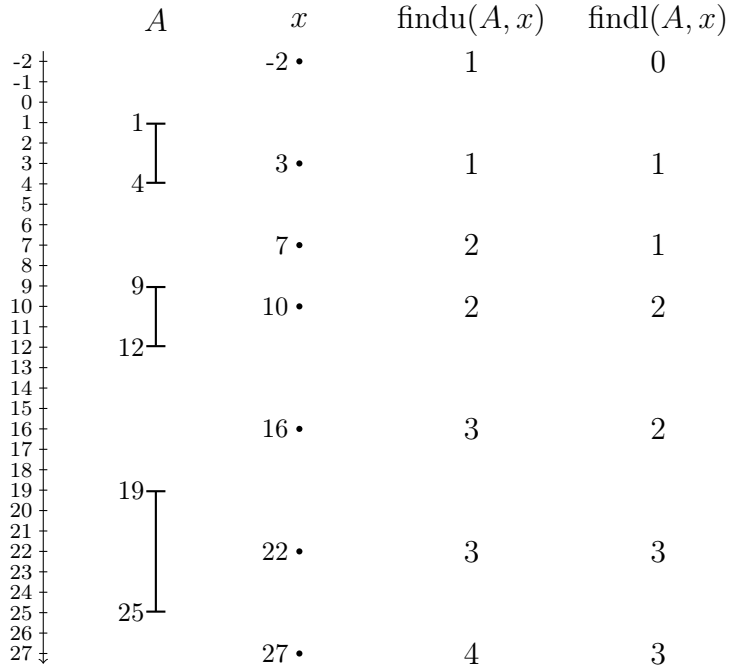

Abbildung 7: Anwendung der Funktionen findl und findu auf die Intervallmenge $A = \{[1, 4], [9, 12], [19, 25]\}$.

Für einen automatischen Algorithmus eignet sich nicht erst die explizite Erstellung der Menge M^x, für Intervalle mit hoher Kardinalität, wie es beispielsweise bei sekundengenauen Zugfolgezeiten der Fall wäre, da hier sehr lange Listen von Zahlen entstehen würden. Es ist somit von Vorteil die Menge M^{xI} aus M iterativ zu entwickeln, in dem stets ein neues Intervall aus M in das Ergebnis eingearbeitet (vereinigt) wird.

Die Intervallmenge M kann ungeordnet sein und selbstverständlich nicht-disjunkte Intervalle enthalten. Um auf spezielle Elemente einer Sequenz $A = (M, \leq)$ zuzugreifen, benutzt man die implizite Indexschreibweise $A = \langle a_1, \ldots, a_n \rangle$.

Wir benötigen zunächst noch einige Hilfsfunktionen. Für den minimalen Index in einer partiell geordneten Intervallmenge $A = \langle a_1, \ldots, a_n \rangle$ mit $a_i = [l_i, u_i]$ $(i \in \{1, \ldots, n\})$ in Bezug auf eine Zahl x (obere Schranke), verwenden wir folgende Funktion

$$\text{findu}(A, x) := \min_{i \in \{1,\ldots,n\}} (\{i \mid x \leq u_i + 1\} \cup \{n + 1\}) \tag{2}$$

Für den respektiven maximalen Index in Bezug auf eine Zahl x (untere Schranke), verwenden wir folgende Funktion

$$\text{findl}(A, x) := \max_{i \in \{1,\ldots,n\}} (\{i \mid x \geq l_i - 1\} \cup \{0\}) \tag{3}$$

Die beiden Funktionen werden in folgendem Beispiel exemplarisch untersucht.

Beispiel 19. Sei $A = \langle a_1, a_2, a_3 \rangle$ eine partiell geordnete Menge mit

$$\begin{aligned} a_1 &= [1, 4], \\ a_2 &= [9, 12], \\ a_3 &= [19, 25]. \end{aligned}$$

Dann ist

- $\text{findu}(A, 3) = 1$, weil $\min_{i \in \{1,2,3\}}(\{i \mid 3 \leq u_i + 1\} \cup \{4\}) = \min\{1, 2, 3, 4\} = 1$
- $\text{findu}(A, 10) = 2$, weil $\min_{i \in \{1,2,3\}}(\{i \mid 10 \leq u_i + 1\} \cup \{4\}) = \min\{2, 3, 4\} = 2$
- $\text{findu}(A, -2) = 1$, weil $\min_{i \in \{1,2,3\}}(\{i \mid -2 \leq u_i + 1\} \cup \{4\}) = \min\{1, 2, 3, 4\} = 1$
- $\text{findu}(A, 27) = 4$, weil $\min_{i \in \{1,2,3\}}(\{i \mid 27 \leq u_i + 1\} \cup \{4\}) = \min\{4\} = 4$
- $\text{findl}(A, 7) = 1$, weil $\max_{i \in \{1,2,3\}}(\{i \mid 7 \geq l_i - 1\} \cup \{0\}) = \max\{0, 1\} = 1$
- $\text{findl}(A, 22) = 3$, weil $\max_{i \in \{1,2,3\}}(\{i \mid 22 \geq l_i - 1\} \cup \{0\}) = \max\{0, 1, 2, 3\} = 3$
- $\text{findl}(A, 27) = 3$, weil $\max_{i \in \{1,2,3\}}(\{i \mid 27 \geq l_i - 1\} \cup \{0\}) = \max\{0, 1, 2, 3\} = 3$

Alle Daten sind auch in Abb. 7 veranschaulicht.

Des Weiteren benötigt man eine Ersetzungsfunktion, die wie folgt definiert ist für die Indizes i, j und einem Intervall I

$$\text{replace}(\langle a_1, \ldots, a_n \rangle, i, j, I) := \langle a_1, \ldots, a_{i-1}, I, a_{j+1}, \ldots, a_n \rangle \tag{4}$$

Für die Veranschaulichung der Funktion in (4) dient folgendes Beispiel.

Beispiel 20. Sei $A = \langle [1, 4], [9, 12], [19, 25] \rangle$ die partiell geordnete Menge aus Beispiel 19. Dann ergibt sich für die Ersetzungsfunktion folgende Beispiele:

$$\begin{aligned} \text{replace}(A, 2, 2, [8, 15]) &= \langle [1, 4], [8, 15], [19, 25] \rangle \\ \text{replace}(A, 1, 2, [1, 15]) &= \langle [1, 15], [19, 25] \rangle \\ \text{replace}(A, 2, 3, [5, 25]) &= \langle [1, 4], [5, 25] \rangle \\ \text{replace}(A, 4, 4, [28, 30]) &= \langle [1, 4], [9, 12], [19, 25], [28, 30] \rangle \\ \text{replace}(A, 2, 1, [6, 7]) &= \langle [1, 4], [6, 7], [9, 12], [19, 25] \rangle \end{aligned}$$

Vor allem das letzte Beispiel ist interessant, da wir mittels $j = i - 1$ auch Intervalle zwischen bestehenden Intervallen hinzufügen können. Dies wird im Algorithmus von Nöten sein, wenn sich die bestehenden Intervalle nicht mit dem hinzuzufügenden Intervall schneiden.

Die Vereinigung einer partiell geordneten Intervallmenge spiegelt sich in folgendem Algorithmus unite wider.

Algorithm 1 Vereinigung Intervalle unite

Input: Intervallmenge M
Output: Geordnete Vereinigung $(M^{xI}, \leq)$

```
1: A ← ⟨⟩                                          // leere, iterative Lösung initialisieren
2: for all [l, u] ∈ M do
3:    i ← findu(A, l)
4:    if i < |A| + 1 then
5:       [l_i, _] ← a_i
6:    else
7:       l_i ← ∞
8:    end if
9:    j ← findl(A, u)
10:   if i > 0 then
11:      [_, u_j] ← a_j
12:   else
13:      u_j ← −∞
14:   end if
15:   I ← [min{l, l_i}, max{u, u_j}]
16:   A ← replace(A, i, j, I)
17: end for
18: return A                                       // A = (M^{xI}, ≤)
```

Da der Algorithmus 1 unite linear über alle Elemente aus M iteriert und die Funktionen findl und findu ebenfalls linear zur Kardinalität von M sind, hat der gesamte Algorithmus eine quadratische Laufzeitkomplexität mit $\mathcal{O}(|M|^2)$.

Satz 21 (Korrektheit und Vollständigkeit (ohne Beweis))**.** *Sei $M \subset \mathcal{I}$ eine Intervallmenge. Dann gilt*

$$(M^{xI}, \leq) = \text{unite}(M).$$

Exemplarisch soll die Korrektheit des Algorithmus in folgendem Beispiel gezeigt werden.

Beispiel 22. Sei $M = \{[1,4],[19,25],[9,12],[10,22]\}$ eine Intervallmenge. Wenden wir Definition 17 auf M an erhalten wir.

$$(M^{xI}, \leq) = \langle [1,4],[9,25] \rangle, \tag{5}$$

weil

$$\begin{aligned} M^{xI} &= \{[1,4],[19,25],[9,12],[10,22]\}^{xI} \\ &\overset{\text{Def. 10}}{=} \{1,\ldots,4,9,\ldots,25\}^{I} \\ &\overset{\text{Def. 12}}{=} \{[1,4],[9,25]\} \end{aligned}$$

Veranschaulicht ist die Vereinigung in Abb. 2. Wenden wir Algorithmus 1, initialisieren wir $A = \langle\rangle$ und erhalten wir folgende Iterationen:

1.
   - $A = \langle\rangle$ am Anfang der Iteration, $[l, u] = [1, 4]$
   - $i = \text{findu}(\langle\rangle, 1) = 1$, $l_1 = \infty$
   - $j = \text{findl}(\langle\rangle, 4) = 0$, $u_0 = -\infty$
   - $I = [\min\{1, \infty\}, \max\{4, -\infty\}] = [1, 4]$
   - $A = \text{replace}(\langle\rangle, 1, 0, [1, 4]) = \langle[1, 4]\rangle$
2.
   - $A = \langle[1, 4]\rangle$ am Anfang der Iteration, $[l, u] = [19, 25]$
   - $i = \text{findu}(\langle[1, 4]\rangle, 19) = 2$, $l_2 = \infty$
   - $j = \text{findl}(\langle[1, 4]\rangle, 25) = 1$, $u_1 = 4$
   - $I = [\min\{19, \infty\}, \max\{25, 4\}] = [19, 25]$
   - $A = \text{replace}(\langle[1, 4]\rangle, 2, 1, [19, 25]) = \langle[1, 4], [19, 25]\rangle$
3.
   - $A = \langle[1, 4], [19, 25]\rangle$ am Anfang der Iteration, $[l, u] = [9, 12]$
   - $i = \text{findu}(\langle[1, 4], [19, 25]\rangle, 9) = 2$, $l_2 = 19$
   - $j = \text{findl}(\langle[1, 4], [19, 25]\rangle, 12) = 1$, $u_1 = 4$
   - $I = [\min\{9, 19\}, \max\{12, 4\}] = [9, 12]$
   - $A = \text{replace}(\langle[1, 4], [19, 25]\rangle, 2, 1, [9, 12]) = \langle[1, 4], [9, 12], [19, 25]\rangle$
4.
   - $A = \langle[1, 4], [9, 12], [19, 25]\rangle$ am Anfang der Iteration, $[l, u] = [10, 22]$
   - $i = \text{findu}(\langle[1, 4], [9, 12], [19, 25]\rangle, 10) = 2$, $l_2 = 9$
   - $j = \text{findl}(\langle[1, 4], [9, 12], [19, 25]\rangle, 22) = 3$, $u_3 = 25$
   - $I = [\min\{9, 10\}, \max\{22, 25\}] = [9, 25]$
   - $A = \text{replace}(\langle[1, 4], [9, 12], [19, 25]\rangle, 2, 3, [9, 25]) = \langle[1, 4], [9, 25]\rangle$

Somit erhalten wir wünschenswerterweise das selbe Ergebnis wie in (5).

4 Rechenergebnisse

Die Berechnungen werden auf einem Intel® Core™ i7-4790K CPU mit 32 GB RAM durchgeführt. Der MIP-Solver ist *Gurobi 7.0.1* [3]. Ziel der Optimierung ist neben der Maximierung der Anzahl einzulegender Trassen der sogenannte Beförderungszeitquotient (BFQ) [2]. Dieser berechnet sich aus dem Quotienten der tatsächlichen Transportzeit zur schnellsten Transportzeit und ist somit stets größer gleich eins. Hat beispielsweise eine eingelegte Systemtrasse einen BFQ von zwei, ist die Transportzeit doppelt so groß im Vergleich zur schnellst möglichen Trasse.

Die Testfälle bestehen aus zwei verschiedenen Systemtrassenanforderungen: Mannheim–Basel und Fulda–Frankfurt. Beide Szenarien basieren auf realen Daten, die für

wissenschaftliche Zwecke der DB Netz AG (Abteilung Langfristfahrplanung) zur Verfügung gestellt wurden. [7]

Beide Szenarien wurden zum einen mit und ohne der Intervallvereinigung gerechnet. Die Rechenergebnisse werden in den entsprechenden, folgenden Kapiteln beschrieben.

4.1 Testfall Mannheim–Basel

Dieser Testfall [2] stellt aus unseren bisherigen Berechnungen meist den kompliziertesten Bereich im deutschen Eisenbahnnetz in Bezug auf die automatische Konstruktion von Systemtrassen dar [9]. Dieser beinhaltet eine Zugcharakteristik und besteht in der Basisvariante (ohne Intervallvereinigung) aus 112 734 Mindestzugfolgezeiten zu vorkonstruierten Trassen. Vereinigt man die resultierenden Intervalle mittels Algorithmus 1 resultiert das in 79 518 Mindestzugfolgerestriktionen. Dies stellt eine Reduktion von ungefähr 30 % dar.

Gibt man in beiden Optimierungsinstanzen (mit und ohne Intervallvereinigung) dem Solver 5 h Rechenzeit führt dies zu folgenden Ergebnissen:

- beide liefern einen durchschnittlichen BFQ von 1,30
- Anzahl neu eingelegter Trassen ohne Vereinigung (mehr Mindestzugfolgerestriktionen): 81
- Anzahl neu eingelegter Trassen mit Vereinigung (weniger Mindestzugfolgerestriktionen): 82

Wir erkennen also, dass eine Trasse mehr bei gleicher Qualität und gleicher Rechenzeit konstruiert werden konnte. Der minimale Overhead bei der Anwendung von Algorithmus 1 (ungefähr 15 s) ist dabei vernachlässigbar klein im Vergleich zur Optimierungszeit.

4.2 Testfall Fulda–Frankfurt

Dieser Testfall ist kleiner als der zuvor berechnete. Für eine Einordnung folgen einige Daten mit und ohne Intervallvereinigung, sowie zur Rechenzeit:

- Anzahl Zugcharakteristika: 3
- Anzahl Mindestzugfolgerestriktionen zu vorkonstruierten Trassen (SPV): 32 276
- Anzahl Mindestzugfolgerestriktionen nach Vereinigung der Intervalle: 16 974 (um $\approx 47\%$ verringert)
- Rechenzeit vorgegeben für Instanz mit und ohne Vereinigung: 15 min

Wendet man den Optimierungssolver wieder auf beide Instanzen (mit und ohne Intervallvereinigung) an, werden die Ergebnisse sogar besser als im zuvor beschriebenen Testfall:

- durchschnittlicher BFQ ohne Vereinigung: 1,20
- durchschnittlicher BFQ mit Vereinigung: 1,16
- Anzahl eingelegte Trassen ohne Vereinigung (mehr Mindestzugfolgerestriktionen): 64
- Anzahl eingelegte Trassen mit Vereinigung (weniger Mindestzugfolgerestriktionen): 85

Es ist offensichtlich, dass mit dem neuen Ansatz (Vereinigung der Mindestzugfolgerestriktionen) sogar deutlich mehr Trassen (21) eingelegt werden konnten und zusätzlich die durchschnittliche Qualität (BFQ) verbessert werden konnte.

Gerade an diesem Testfall sieht man, dass sich eine Vereinigung lohnt. Es folgt eine Betrachtung für ein konkretes zu konstruierendes Trassenbündel mit folgenden Lösungsdaten im Überblick:

- 3 zu konstruierende Systemtrassen
- BFQ mit und ohne Vereinigung: 1,0
- siehe Abb. 8 für Entwicklung des Zielfunktionswerts

Es lässt sich schlussfolgern: Die Vereinigung der Mindestzugfolgezeiten kommt in kürzerer Zeit, bereits nach 4 s, zum selben Ergebnis. Dies spricht für die Äquivalenz der Ansätze (gibt man genügend Rechenzeit, werden beide Verfahren die selben Ergebnisse liefern), allerdings scheint die Vereinigung der Mindestzugfolgezeiten schneller zu diesem Ergebnis zu kommen. Dies kann im besten Fall zu einer größeren Anzahl an konstruierten Systemtrassen resultieren, wie es sich in den Testfällen auch widerspiegelt.

5 Zusammenfassung und Ausblick

In dieser Arbeit wurde die Anzahl der Mindestzugfolgezeiten versucht zu minimieren, in dem sich überlappende Bereiche mittels ganzzahliger Intervallvereinigung zusammengeführt werden. Nach der mathematischen Einführung der Grundbegriffe, wurde ein Algorithmus zum genannten Thema vorgestellt. Die erfolgreiche Anwendung und Verbesserung auf reale Szenarien konnte gezeigt werden. Diese beinhalten zum einen eine reduzierte Netzwerkgröße (Instanzgröße), die sich zum anderen in eine höheren Anzahl an eingelegten Systemtrassen und teilweise besserer Qualität selbiger widerspiegeln.

Allerdings lässt die vorgestellte Methode noch Raum für weitere Forschung. Ein alternativer Ansatz wäre die Intervallmenge M vorher zu sortieren mittels $(M, \leq)$ und anschließend könnte man mittels eines linearen Algorithmus die Intervalle zusammenfassen. Somit hätte man eine Laufzeitkomplexität aus der Summe des Sortieralgorithmus' und der linearen Zusammenfassung. Hier würde sich eine Überprüfung anhand von Mengen M mit hoher Kardinalität lohnen. Da für diese Anwendung, die die Minimierung der Anzahl der Mindestzugfolgezeiten darstellt, nicht auftreten, bestand bisher nicht der

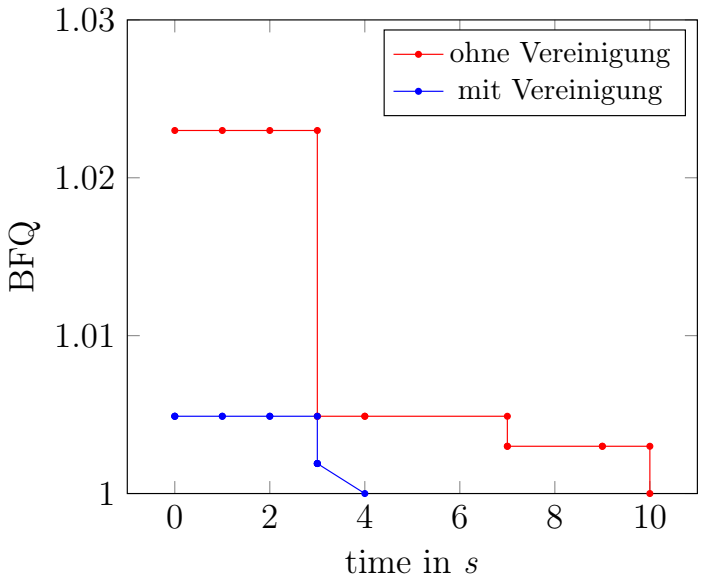

Abbildung 8: Entwicklung des Zielfunktionswerts bei der Konstruktion eines Trassenbündels.

Bedarf, da die Rechenzeiten sehr gering im Verhältnis zu den folgenden Optimierungsproblemen sind [2].

Abschließend lässt sich zusammenfassen, dass die Anwendung dieser Methode stets zu gleich guten oder besseren Ergebnissen führt. Es ist somit ratsam diesen Ansatz in jeder Form der automatischen Fahrplanung einzusetzen, wenn fixierte Mindestzugfolgezeiten im System vorhanden sind.

Literatur

[1] V. Cacchiani, L. Galli, and P. Toth. A tutorial on non-periodic train timetabling and platforming problems. In *EURO Journal on Transportation and Logistics*, volume 4(3), pages 285–320. Springer, 2015.

[2] Peter Großmann, Alexander Labinsky, Jens Opitz, and Reyk Weiß. Capacity-utilized integration and optimization of rail freight train paths into 24 hours timetables. In *MT-ITS*, pages 389–396. TUDpress, 2013.

[3] Zonghao Gu, Edward Rothberg, and Robert Bixby. *Gurobi 7.0.1*. Gurobi Optimization, Inc., Houston, TX, May 2015.

[4] Michael Kümmling, Peter Großmann, Karl Nachtigall, Jens Opitz, and Reyk Weiß. A state-of-the-art realization of cyclic railway timetable computation. *Public Transport*, 7(3):281–293, 2015.

[5] Karl Nachtigall. *Periodic Network Optimization and Fixed Interval Timetable*. Habilitation thesis, University Hildesheim, 1998.

[6] Jens Opitz. *Automatische Erzeugung und Optimierung von Taktfahrplänen in Schienenverkehrsnetzen*. PhD thesis, Reihe: Logistik, Mobilitöät und Verkehr. Gabler Verlag | GWV Fachverlage GmbH, 2009.

[7] Daniel Pöhle and Werner Weigand. Neue strategien für die konstruktion von systemtrassen in einem industrialisierten fahrplan. In *VWT Dresden, Presentation*, 2014.

[8] L. E. Ward. Partially ordered topological spaces. *Proc. Amer. Math. Soc.*, 5:144–161, 1954.

[9] Reyk Weiß, Jens Opitz, and Karl Nachtigall. A novel approach to strategic planning of rail freight transport. In *Operations Research Proceedings 2012*, pages 463–468. Springer, 2012.