Centrum voor Wiskunde en Informatica

Spoorwegdienstregelingontwikkeling

A. Schrijver
A. Steenbeek

1993

Inhoud

1. Inleiding 1
2. Het model 3
3. De methode 6
4. Het testen van de constraints 10
5. Minimalisering van de wachttijden 13
6. Variabiliteit van de rijtijden 16
7. Conclusies 19

1. Inleiding

In de eerste fase van het project Dienstregeling voor Nederlandse Spoorwegen N.V. is een methode ontwikkeld welke een dienstregeling ontwerpt, op basis van een gegeven dataset van in te leggen treinen met zekere gewenste aansluitingen en frequenties.

De in te zetten treinen werden beperkt tot het zgn. 'Rompnet', terwijl alleen het uurlijkse dienstregelingspatroon diende te worden bepaald.

In de inputfile `etappes` wordt elke in te leggen uurlijkse trein beschreven door middel van de achtereenvolgens af te leggen etappes, de rijtijd voor elk der etappes, en de minimale en maximale stoptijd tussen elk tweetal opvolgende etappes.

De inputfile `marktcon` geeft de marktconstraints. Deze geven aan welke treinen op elkaar dienen aan te sluiten (waarbij de minimale en maximale overstaptijd wordt gegeven) en welke treinen (of treinetappes) elkaar om het halfuur dienen op te volgen. In sommige gevallen wordt een nog fijnere frequentie voorgeschreven (met marges: bijvoorbeeld 13-17, 43-47). Bovendien staan in `marktcon` een aantal 'absolute' eisen, i.h.b. vastliggende vertrektijden van internationale treinen.

Tenslotte geeft de inputfile `conflict` alle conflictsituaties. Deze beschrijft de minimaal geëiste opvolgingstijd op gemeenschappelijke trajecten van treinen. Evenzo worden hier de beperkingen beschreven veroorzaakt door kruisingen en enkelsporige trajecten.

Tot de resultaten van de eerste fase behoren:

(i) De opstelling van een combinatorisch optimaliseringsmodel, waarbinnen de gestelde eisen alle worden beschreven;

(ii) Het ontwikkelen van een algoritme en van programmatuur die een dienstregeling ontwerpen welke voldoet aan alle eisen, mits deze geen tegenstrijdigheden bevatten;

(iii) Het ontwikkelen van programmatuur welke een eerste test uitvoert op de consistentie van de constraints, en een lijst geeft van minimale collecties tegenstrijdige constraints (deze test is geschreven in de taal perl, maar omzetting naar C++ geeft een aanzienlijk versnelling);

(iv) De uitbreiding van het model tot minimalisering van stop- en overstaptijden.

2. Het model

De input van het probleem bestaat als gezegd uit drie files:

`etappes`

`marktcon`

`conflict`

Deze worden als volgt omgezet naar het model.

Elke te bepalen vertrektijd wordt voorgesteld door een variabele. De variabele v_t stelt de vertrektijd van treinetappe t voor. Dus de regels:

trno	etno	soort	van	naar	duur	wachttijd
38	00	IR	RTD	RTA	8	1-5
38	01	IR	RTA	GD	7	1-5
38	02	IR	GD	WD	8	0
38	03	IR	WD	UT	9	2-5
38	04	IR	UT	ED	21	0
38	05	IR	ED	AH	9	4-5
38	06	IR	AH	NM	11	

in de file `etappes` geven de variabelen $v_{3800}, v_{3801}, v_{3802}, v_{3803}, v_{3804}, v_{3805}, v_{3806}$. Hierin stelt bijvoorbeeld v_{3803} de te bepalen vertrektijd (modulo 60 minuten) van etappe 03 van trein(serie) 38 voor, kortweg treinetappe 3803.

Tussen elk tweetal opvolgende etappes van één trein wordt het verband tussen de vertrektijden op de volgende manier aangegeven. Treinetappe 3801 heeft een rijtijd van 7 minuten en vervolgens een stoptijd van tussen 1 en 5 minuten. Er moet dus gelden:

$$8 \leq v_{3802} - v_{3801} + 60p_{3801,3802} \leq 12$$

waarbij $p_{3801,3802}$ een geheel getal is. Hiermee wordt dus bereikt dat het verschil van de aankomsttijd van etappe 3801 in Utrecht en de vertrektijd van de volgende etappe 3802 in het interval $[2, 5]$ ligt, *op een geheel veelvoud van 60 minuten na.*

Ook de marktconstraints kunnen op deze manier worden beschreven. De regel

```
AD      3805    3355    2-5
```

in de file `marktcon` betekent dat een aansluiting moet bestaan van treinetappe 3805 op treinetappe 3355 (in Arnhem), met een minimale overstaptijd van 2 minuten en een maximale overstaptijd van 5 minuten. Aangezien de rijtijd van treinetappe 3805 gelijk is aan 9 minuten, krijgen we de volgende constraint:

(1)
$$11 \leq v_{3355} - v_{3805} + 60p_{3805,3355} \leq 14.$$

Wederom is de p-variabele geheeltallig, zodat het verschil van de vertrektijden wordt bepaald tot op gehele veelvouden van 60 minuten na.

Evenzo kunnen de frequentieconstraints worden behandeld. De regel:

```
DD      3800    3900    30
```

in de file `marktcon` betekent dat de vertrektijd van treinetappe 3800 precies 30 minuten moet verschillen van die van 3900. Dit geeft als constraint:

$$30 \leq v_{3900} - v_{3800} + 60p_{3800,3900} \leq 30.$$

De regel:

```
DD      16200   14900   13-17, 43-47
```

in de file `marktcon` betekent dat het verschil van de vertrektijden van treinetappes 16200 en 14900 in een van de intervallen 13-17,43-47 moet liggen. Dit kan worden geformuleerd als: het verschil moet *zowel* in het interval 13-47 *als* in het interval 43-77 liggen (beide modulo 60). We krijgen dus twee constraints:

$$13 \leq v_{14900} - v_{16200} + 60p'_{16200,14900} \leq 47,$$
$$43 \leq v_{14900} - v_{16200} + 60p''_{16200,14900} \leq 77,$$

waarbij zowel de p'- als de p''-variabele geheeltallig is. De marktconstraint

```
ABS      100      0        8
```

schrijft de absolute vertrektijd (modulo 60) van treinetappe 100 voor. Dit kan worden beschreven door het invoeren van een 'dummy' trein 0, waarvan de vertrektijd v_0 achteraf op 0 wordt gesteld. Dus geeft de bovenstaande marktconstraint:

$$8 \leq v_{100} - v_0 + 60p_{0,100} \leq 8.$$

De file `conflict` beschrijft alle te vermijden conflictsituaties. Een voorbeeld van een regel uit deze file is:

```
GD-WD    3802    402     59-64
```

Dit betekent dat de vertrektijd van treinetappe 402 minus de vertrektijd van treinetappe 3802 niet in het interval 59-64 mag liggen (modulo 60), vanwege de voorgeschreven minimale opvolgingstijd op het traject Gouda-Woerden. Het verschil moet dus in het interval 5-58 (modulo 60) liggen. We krijgen dan als ongelijkheid:

$$5 \leq v_{402} - v_{3802} + 60p_{3802,402} \leq 58.$$

3. De methode

Het probleem is nu volledig vertaald in het volgende wiskundige probleem. Gegeven zijn m ongelijkheden:

$$a_j \leq v_{i_j} - v_{i'_j} + 60p_j \leq b_j \qquad (j = 1, \ldots, m) \tag{2}$$

in de variabelen $v_1, \ldots, v_n$ en $p_1, \ldots, p_m$. Bepaal een oplossing voor deze ongelijkheden waarbij $v_1, \ldots, v_n, p_1, \ldots, p_m$ geheeltallig zijn.

Dit is in het algemeen een lastig probleem. Als de eis tot geheeltalligheid niet gesteld zou worden, dan zou het probleem zeer snel zijn op te lossen met de methoden van de lineaire programmering. Maar vanwege de geheeltalligheidseis wordt het een stuk moeilijker, ook omdat sprake is van een groot aantal variabelen: voor het onderzochte NS-probleem geldt $n = 537$ en $m = 4681$.

Hierbij is n van de orde van het aantal regels in de file `etappes` en m van de orde van het aantal regels in de files `etappes`, `marktcon` en `conflict` tezamen.

Er zijn echter een aantal methodieken toepasbaar welke het probleem aanzienlijk reduceren. Allereerst kan worden gesteld dat het alleen van belang is te zorgen dat de p-variabelen geheeltallig worden. Bewezen kan worden dat als er een oplossing v, p is waarbij alleen het p-gedeelte geheeltallig is, dan kan deze oplossing op eenvoudige manier zo worden gewijzigd dat ook het v-gedeelte geheeltallig wordt. Dit berust op de zgn. 'totale unimodulariteit' van de onderliggende matrix.

Dit maakt dat de 'bottleneck' van het probleem bepaald wordt door het aantal p-variabelen. Hoe meer we dit aantal kunnen reduceren, des te sneller is het probleem oplosbaar.

Een eerste reductie is als volgt. In veel van de ongelijkheden (2) blijkt dat $a_j = b_j$ geldt. Dit doet zich bijvoorbeeld voor als voor twee opvolgende treinetappes een vaste stoptijd is voorgeschreven of als twee treinen precies om het halfuur moeten gaan rijden.

In dat geval kan de variabele p_j worden geëlimineerd, en hetzelfde geldt voor één van de twee variabelen v_{i_j} en $v_{i'_j}$. Toegepast op het NS-probleem geeft dit een reductie tot 3732 constraints met 235 v-variabelen en 3732 p-variabelen.

Een andere reductie onstaat door samenvoeging van constraints (2) waarvoor de v-variabelen hetzelfde zijn. Als in (2) bijvoorbeeld de volgende twee ongelijkheden voorkomen:

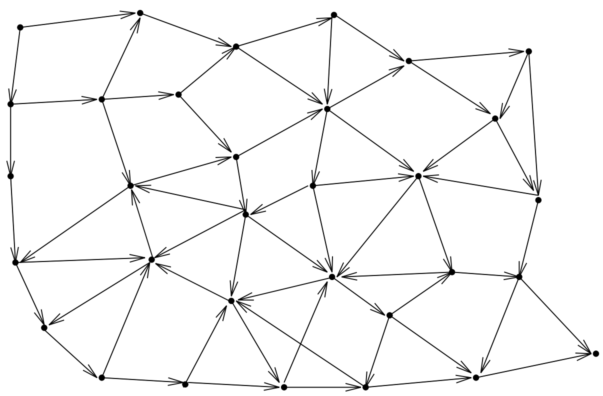
$$3 \leq v_i - v_{i'} + 60p_j \leq 25,$$
$$16 \leq v_i - v_{i'} + 60p_{j'} \leq 44,$$

dan kunnen deze worden vervangen door:

$$16 \leq v_i - v_{i'} + 60p_j \leq 25. \tag{3}$$

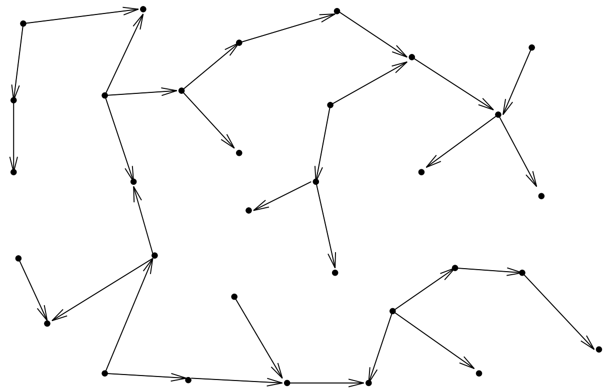
Dit spaart dus een constraint en een p-variabele uit. Toegepast op het NS-probleem geeft dit een reductie tot 1274 constraints met 235 v-variabelen en 1274 p-variabelen.

Een volgende reductie wordt verkregen op de volgende manier. We kunnen de constraints (2) representeren door middel van een gerichte graaf $G = (V, A)$. De punten van deze graaf worden gevormd door de variabelen v_i en de pijlen worden gerepresenteerd door de variabelen p_j. De variabele p_j geeft dan een pijl van $v_{i'_j}$ naar v_{i_j}.



een graaf G

We krijgen op deze manier een graaf G met n punten en m pijlen, welke in het algemeen samenhangend zal zijn. (Als dit niet het geval is, dan is het probleem te decomponeren in kleinere problemen.)



Een opspannende boom T in G

We kunnen dan een 'opspannende boom' T in G vinden. Dit is een deelgraaf die weer samenhangend is en die bovendien geen circuits omvat. Het kan nu met behulp van eenvoudige grafentheorie worden aangetoond dat we de p-variabelen overeenkomend met de kanten in T gelijk mogen stellen aan 0, zonder afbreuk te doen aan de oplosbaarheid van het systeem. Dit geeft een reductie tot 1040 p-variabelen.

Met behulp van deze en enige andere technieken is het mogelijk het probleem nog verder te reduceren. Daardoor wordt het mogelijk de constraints op te lossen binnen redelijk snelle tijd, met behulp van geheeltallige programmeringsmethoden. We hebben hiertoe het pakket CPLEX gebruikt.

CPLEX kent een aantal parameters welke het zoekproces sturen en daardoor de looptijd beïnvloeden. Deze parameters beschrijven de door CPLEX te volgen splitsing van variabelen. Bij uittesten van verschillende splitsvolgordes bleek CPLEX in staat te zijn in 44 CPUseconden (op een R4400) een toegelaten dienstregeling te bepalen. We geven een voorbeeld van een dergelijke dienstregeling in de bijlage.

De tijd waarin de dienstregeling wordt gevonden is sterk afhankelijk van de juiste keuze van de splitsvolgorde, en verder onderzoek zal moeten uitwijzen hoe deze volgorde snel kan worden gevonden.

Daarnaast blijkt dat CPLEX in staat is in vrij korte tijd (ongeveer 5 CPUminuten) een flink aantal verschillende dienstregelingen te genereren. Deze worden gegenereerd door CPLEX een aantal verschillende zoekstrategieën te laten volgen.

4. Het testen van de constraints

De bovenbeschreven methode maakt het ook mogelijk een test uit te voeren op de consistentie van de constraints, en eventuele tegenstrijdigheden op te sporen. Er zijn drie niveaus van tegenstrijdigheden:

(i) tegenstrijdigheden afkomstig van 'te krappe' cykels in de graaf;

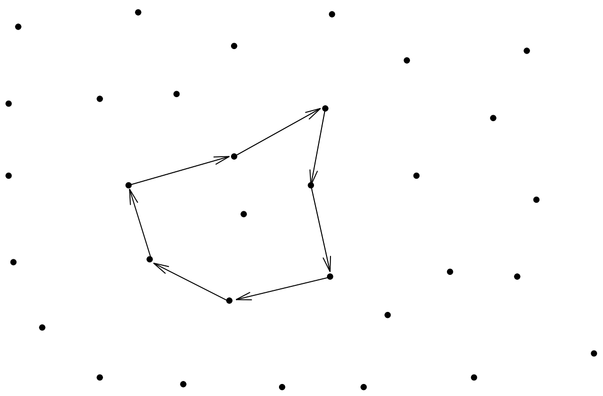
(ii) tegenstrijdigheden afkomstig van lineaire combinaties van constraints;

(iii) combinatorische tegenstrijdigheden.

Het eerste niveau van tegenstrijdigheden wordt op de volgende manier gevormd. Neem aan dat we de volgende constraints hebben:

$$
\begin{aligned}
a_1 &\le v_{i_2} - v_{i_1} + 60p_1 \le b_1,\\
a_2 &\le v_{i_3} - v_{i_2} + 60p_2 \le b_2,\\
&\vdots\\
a_k &\le v_{i_1} - v_{i_k} + 60p_k \le b_k.
\end{aligned} \tag{4}
$$

Dit komt overeen met een cykel in de graaf G: $v_{i_1} \to v_{i_2} \to \cdots \to v_{i_k}$:



Een cykel in G

We kunnen nu de constraints in (4) bij elkaar optellen. Doordat de v-variabelen tegen elkaar wegvallen krijgen we:

$$a_1 + a_2 + \cdots + a_k \leq 60(p_1 + p_2 + \cdots + p_k) \leq b_1 + b_2 + \cdots + b_k, \tag{5}$$

oftewel:

$$\frac{a_1 + a_2 + \cdots + a_k}{60} \leq p_1 + p_2 + \cdots + p_k \leq \frac{b_1 + b_2 + \cdots + b_k}{60}. \tag{6}$$

Vanwege het feit dat de p-variabelen geheeltallig zijn, kunnen we het linkerlid in (6) naar boven afronden en het rechterlid naar beneden:

$$\lceil\frac{a_1 + a_2 + \cdots + a_k}{60}\rceil \leq p_1 + p_2 + \cdots + p_k \leq \lfloor\frac{b_1 + b_2 + \cdots + b_k}{60}\rfloor. \tag{7}$$

Als nu het linkerlid *groter* is dan het rechterlid, dat wil zeggen, als:

$$\lceil\frac{a_1 + a_2 + \cdots + a_k}{60}\rceil > \lfloor\frac{b_1 + b_2 + \cdots + b_k}{60}\rfloor, \tag{8}$$

dan hebben we een contradictie in de constraints verkregen: een te krappe cykel.

In principe zouden we voor alle mogelijke cykels in de graaf G kunnen testen of (8) geldt. Maar het aantal cykels is veel te groot om deze alle stuk voor stuk langs te gaan.

In de aan NS geleverde software wordt een gedeelte van de cykels getest. Deze hebben een hoge kans om te krap gekozen te zijn als de constraints tegenstrijdig zijn.

De software geeft een overzicht van alle gevonden te krappe cykels. Het programma is geschreven in de taal perl, welke geschikt is voor snel te beschrijven datamanipulaties, maar in performance-tijd langzamer is dan C++. Omzetting naar C++ zal een aanzienlijke versnelling opleveren. Bovendien kan de programmatuur worden uitgebreid tot een methode die (op een impliciete wijze) *alle* cykels test.

Met behulp van het LP-gedeelte van CPLEX kan vervolgens worden nagegaan of de constraints, opgevat als *lineaire* constraints, tegenstrijdigheden bevatten. Hierbij wordt de geheeltalligheidseis voorlopig even buiten beschouwing gelaten.

Ook dit niveau van tegenstrijdigheden is vrij snel te ontdekken, en programmatuur kan ontwikkeld worden welke vrij snel de contradicties aan het licht brengt (in de vorm van minimale verzamelingen onderling tegenstrijdige constraints).

Het niveau van combinatorische tegenstrijdigheden ligt het diepst. Deze kunnen worden ontdekt met behulp van het IP-gedeelte van CPLEX. (IP = integer programming.) Nog onduidelijk is hoe de gevonden contradicties kunnen worden gereduceerd tot minimale collecties onderling tegenstrijdige constraints. Verwacht wordt dat *als* de constraints tegenstrijdig zijn, er een kleine collectie is aan te wijzen, maar hoe snel deze aanwijzing in de praktijk kan geschieden vergt verder onderzoek.

5. Minimalisering van de wachttijden

In het model kan op eenvoudige wijze ook de minimalisering van de stop- en overstaptijden worden opgenomen. Elke gewenste stop- of overstaptijd komt overeen met een een van de constraints

$$a_j \leq v_{i_j} - v_{i'_j} + 60p_j \leq b_j. \tag{9}$$

Willen we nu deze wachttijd minimaliseren, dan komt dit neer op het minimaliseren van

$$v_{i_j} - v_{i'_j} + 60p_j - a_j. \tag{10}$$

Dus als we het totaal van deze stop- en overstaptijden willen minimaliseren, dan dienen we dus de som

$$\sum_{j \in J} (v_{i_j} - v_{i'_j} + 60p_j - a_j) \tag{11}$$

te minimaliseren. Hierin stelt J de verzameling voor van al die indices j welke corresponderen met een te minimaliseren stop- of overstaptijd.

We hebben dan het oorspronkelijke probleem van het vinden van een toegelaten oplossing, veranderd in het probleem van het vinden van een oplossing waarbij (11) wordt geminimaliseerd. Dit is weer een geheeltallig programmeringsprobleem, waarop de in de vorige paragrafen beschreven methode toepasbaar is.

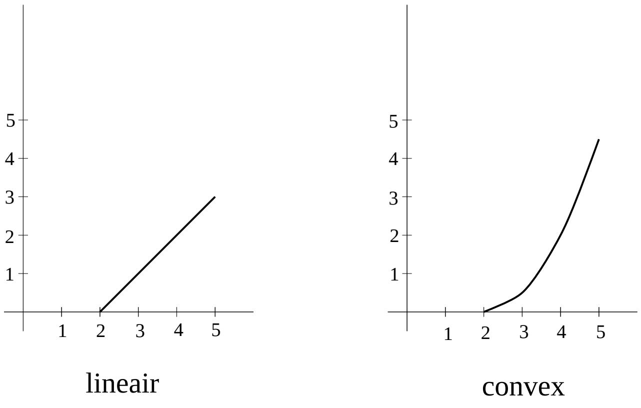
Dit is uit te breiden tot de situatie waar bij elke gewenste stop- of overstaptijd een gewicht wordt gegeven dat het belang aangeeft van de betreffende stop- of overstaptijd. Dit gewicht kan samenhangen met het aantal verwachte wachtende of overstappende passagiers.

Dit wil zeggen dat als we bij elke constraint j weten wat het aantal passagiers c_j is dat belang heeft bij een zo kort mogelijke wachttijd, dan kunnen we minimaliseren:

$$\sum_{j=1}^{m} c_j (v_{i_j} - v_{i'_j} + 60p_j - a_j). \tag{12}$$

Ook hierop zijn de methoden van de geheeltallige programmering toepasbaar.

Vaak is het echter gewenst dat een wachttijd niet lineair wordt gewaardeerd, maar eerder convex.



Ruwweg gezegd wil dit zeggen dat de bereidheid tot een minuut langer wachten afneemt naarmate men langer heeft gewacht. De laatste wachtminuten voelen langer aan dan de eerste.

Dus als er twee groepen wachtenden zijn, waarvan de ene uit 50 reizigers bestaat en de andere uit 40 reizigers, dan kan het beter zijn de wachttijden enigszins gelijk te verdelen, in plaats van de grotere groep het minimale aantal wachtminuten toe te kennen, ten koste van het geduld van de iets kleinere groep.

Ook deze niet-lineaire, convexe wachttijdwaardering past in het model. Hiertoe moeten een aantal extra variabelen worden ingevoerd. Voor deze variabelen geldt niet de geheeltalligheidseis, dus het geeft geen vergroting van de complexiteit van het probleem.

Evenzo kan de eis van regelmatige opvolging van treinen in het model worden opgenomen. Als de geëiste frequentieligging van twee treinen op hetzelfde traject gegeven wordt door het interval 28-32, dan geeft dit in het model:

$$28 \leq v_{i_j} - v_{i'_j} + 60p_j \leq 32.$$

Als er nu een marktvoorkeur bestaat voor het midden van het interval, dan kunnen we een extra variabele y_j invoeren waarvoor moet gelden:

$$v_{i_j} - v_{i'_j} + 60p_j - y_j \leq 30,$$
$$v_{i_j} - v_{i'_j} + 60p_j + y_j \geq 30.$$

Als we nu y_j minimaliseren, dan zal y_j gelijk zijn aan de afwijking van $v_{i_j} - v_{i'_j} + 60p_j$ met 30. De waarde van y_j geeft dus de afwijking aan tussen de bereikte frequentie en de optimale 30 minuten-frequentie.

Doen we dit voor alle 28-32 constraints, en minimaliseren we de som van alle zo ontstane variabelen y_j, dan worden aldus de regelmaten tegen elkaar afgewogen.

Op een zelfde manier kunnen de regelmaat-constraints van het type 13-17,43-47 worden behandeld. Ook bestaat de mogelijkheid een gewogen som te minimaliseren, afhankelijk van het aantal verwachte reizigers, en dit te combineren met het minimaliseren van de wachttijden.

Er zijn verschillende andere optimaliseringscriteria welke in het model passen, waaronder het afwegen van wachttijden tegen benodigde uitbreidingen van infrastructuur (viaducten, vier-sporige trajecten).

6. Variabiliteit van de rijtijden

Door middel van een kleine uitbreiding van het model is het ook mogelijk de rijtijden per etappe variabel te maken, tussen gegeven onder- en bovengrenzen. Het betekent dat voor elke treinetappe t, naast een vertrektijd v_t ook een aankomsttijd a_t moet worden bepaald.

In dit geval worden in de file `etappes` geen vaste, maar variabele rijtijden gespecificeerd, bijvoorbeeld:

trno	etno	soort	van	naar	duur	wachttijd
38	00	IR	RTD	RTA	8-9	1-5
38	01	IR	RTA	GD	7-8	1-5
38	02	IR	GD	WD	8-9	0
38	03	IR	WD	UT	9-10	2-5
38	04	IR	UT	ED	20-23	0
38	05	IR	ED	AH	9-10	4-5
38	06	IR	AH	NM	10-12	

Daarnaast dient de file `conflict` te worden aangepast. Het zal bijvoorbeeld nodig zijn dat niet alleen vertrektijden een bepaalde onderlinge tijdsafstand hebben, maar ook de aankomsttijden.

Als treinetappe t en treinetappe t' hetzelfde traject berijden en een minimale opvolgingstijd van 3 minuten dienen te hebben, dan geeft dit als constraints:

(13)
$$3 \leq v_{t'} - v_t + 60p \leq 57,$$
$$3 \leq a_{t'} - a_t + 60p' \leq 57.$$

Dergelijke constraints zijn beide nodig, aangezien de rijtijden variabel zijn, en dus de opvolgingstijd niet uitsluitend afhangt van de vertrektijden.

Maar dit is nog niet voldoende. Het kan zijn dat de file `etappes` de volgende rijtijden geeft voor etappes t and t':

$$\text{rijtijd } t\text{: 33-37 minuten}$$
$$\text{rijtijd } t'\text{: 29-32 minuten.}$$

Dit geeft als constraints:

(14)
$$\begin{aligned} 33 \leq a_t - v_t + 60p'' \leq 37, \\ 29 \leq a_{t'} - v_{t'} + 60p''' \leq 32. \end{aligned}$$

De constraints (13) en (14) zijn echter niet voldoende, aannemende dat het betreffende traject niet viersporig is en de twee treinen elkaar dus niet kunnen inhalen.

De volgende waarden geven nl. een oplossing voor (13) en (14):

$$v_t = 0, a_t = 37, v_{t'} = 3, a_{t'} = 32,$$
$$p = 0, p' = 1, p'' = 0, p''' = 0.$$

Dit betekent dat trein t wordt ingehaald door trein t'. Dit had kunnen worden voorkomen bij vaste rijtijden, doordat dan de vertrek-vertrek constraint in (13) kan worden aangepast, en we precies het toegelaten interval kunnen beschrijven dat geen conflict oplevert.

Het probleem kan worden opgelost door als extra constraint toe te voegen:

(15)
$$p - p' - p'' + p''' = 0.$$

In zekere zin kan dit worden geïnterpreteerd als: de volgorde van de treinen t and t' mag niet veranderen op het betreffende traject. Het kan bewezen worden dat toevoeging van de constraints (15) voldoende is om de betreffende verboden inhalingen te voorkomen.

Het is overigens nog een lastig probleem om precies in de file `conflict` te beschrijven welke de verboden intervallen zullen zijn voor paren variabelen. Als elke regel in de file `conflict` afkomstig zou zijn van twee volledige etappes, dan zal het verboden interval eenvoudig te beschrijven zijn. Maar als twee treinetappes slechts gedeeltelijk een gemeenschappelijk traject hebben, bijvoorbeeld tussen Vught aansluiting en 's-Hertogenbosch, dan zou het kunnen zijn dat het conflict alleen te beschrijven is door ook een 'doorlooptijd' in Vught aansluiting te introduceren.

7. Conclusies

1. De beschreven methode is in staat binnen tamelijk korte tijd een dienstregeling op te stellen voor de door NS geleverde files `etappes`, `marktcon`, en `conflict`. Het is waarschijnlijk dat de methode ook voor vergelijkbare input een vergelijkbare 'performance' geeft.

2. De methode is in staat een test uit te voeren op de consistentie van de constraints, en minimale verzamelingen onderling tegenstrijdige constraints op te sporen. Als de tegenstrijdigheid voortkomt uit eenvoudige cykels in de graaf, dan wordt deze ontdekt door het geleverde perl-programma. Deze methode is uit te breiden tot een methode waarbij alle cykels getest worden.

   Als de tegenstrijdigheid voortkomt uit het niet-bestaan van een fractionele oplossing voor het bijbehorende lineaire probleem, dan is ook dit eenvoudig en snel algoritmisch naar voren te brengen. Alle overige tegenstrijdigheden kunnen met behulp van geheeltallige programmering worden ontdekt, maar de efficiëntie hiervan dient onderzocht te worden.

3. In de methode en de programmatuur kunnen o.a. de volgende voorzieningen worden opgenomen:

   (i) Minimalisering van de wachttijden van de reizigers. Dit kan een gewogen minimalisering zijn, op basis van het verwachte aantal reizigers. Ook kan als optimaliseringscriterium de totale rijtijd van een trein worden opgenomen.

   (ii) Optimalisering van de regelmaat van de treinen (bijvoorbeeld zo dicht mogelijk bij het midden van de geëiste 13-17, 43-47 frequenties).

   (iii) Variabiliteit van rijtijden, gegeven door onder- en bovengrenzen.

   (iv) Afweging van op te heffen knelpunten tegen marktconstraints. Het is mogelijk het programma zo uit te breiden dat bij tegenstrijdigheid van constraints een zo klein mogelijke (of zo zuinig mogelijke) verzameling knelpunten wordt gegenereerd, welke dienen te worden opgelost (bijv. door het bouwen van viaducten of het aanleggen van meer sporen) om de gewenste dienstregeling mogelijk te maken. Ook is een afweging mogelijk tussen enerzijds marktconstraints en anderzijds investeringen.