ERASMUS UNIVERSITEIT ROTTERDAM

Erasmus School of Economics

Bachelorscriptie Econometrie en Operationele Research

Een Effectief Peak Period Heuristiek voor Railway Rolling Stock Planning

Naam: F.J. Hamers
Studentnummer: 453777

Begeleider: R.N. van Lieshout
Tweede beoordelaar: dr. T.A.B. Dollevoet

7 juli 2019

Abstract

Om te concurreren in een biedingsproces voor de concessie voor een bepaald gebied, moeten spoorwegmaatschappijen een plan maken op strategisch niveau voor het aantal aan te schaffen treinstellen. In deze scriptie wordt het *train unit assignment problem* (TUAP) opgelost aan de hand van het *Peak Period Heuristiek* gepresenteerd in het onderzoek van Cacchiani et al. (2019). Naast de originele implementatie van het paper, is er een extra versie van het heuristiek bekeken waarbij de mogelijke treinstellen worden gesorteerd op basis van kosten (*Versie Kosten*) in plaats van capaciteit. Aan de hand van vier instanties van verschillende grootte, kan uit de resultaten van deze scriptie worden geconcludeerd dat het heuristiek met *Versie Kosten* (gemiddelde gap van 16.98%) beter presteert dan het origineel (gemiddelde gap van 24.74%). Naast de replicatie van de methodes uit Cacchiani et al. (2019), is een extensie bestudeerd op het TUAP. In deze scriptie is treinuitval meegenomen in het heuristiek, waarbij inzicht wordt gegeven in de trade-off tussen de kosten voor het aantal treinstellen en het service level. Aan de hand van verschillende versies van 'safety stock', is een treinstel uitgevallen en een nieuwe toewijzing van de treinstellen aan de trips bepaald. Dit is per safety stock drie keer herhaald waarbij per keer een ander type treinstel uitvalt. Van deze waardes is een gewogen gemiddelde genomen op basis van de initiële verdeling van treinstellen per type. Daarnaast is een worst-case scenario analyse uitgevoerd om te onderzoeken of een type treinstel significant minder goed presteert op basis van zitplaatstekort. Dit zitplaatstekort bestaat uit de te weinig ingeplande passagiersstoelen voor een trip. Uit de resultaten van deze scriptie kan worden geconcludeerd dat de trade-off tussen treinkosten en service level voorkomt bij het uitvallen van treinen. Aan de hand van een plot tussen de treinkosten en zitplaatstekort, kunnen managers van spoorwegmaatschappijen beslissen welke hoeveelheid safety stock zij prefereren aan de hand van hun optimale trade-off. In deze scriptie is geen eenduidig treinstel gevonden dat het vaakst het worst-case zitplaatstekort bevat.

Trefwoorden— train unit assignment problem, railway rolling stock planning, robuust optimaliseren, safety stock

Inhoudsopgave

1 Introductie

Wanneer op regionaal niveau een spoorwegmaatschappij moet worden geselecteerd voor het vervoer van passagiers, vindt er een *concurrerend biedingsproces* plaats. Om een idee te krijgen van de kosten construeren spoorwegmaatschappijen een plan op strategisch niveau voor het treinvervoer in een bepaald gebied om te concurreren in het biedingsproces. In dit plan wordt gekeken naar het optimaal verdelen van treinstellen over verschillende trips om de aanschafkosten van de treinstellen te minimaliseren en aan de passagiersvraag te voldoen. Het bepalen van een zogeheten efficiënte railway rolling stock is cruciaal om drie redenen. Ten eerste is het aanschaffen van treinstellen een kostbare langetermijninvestering die niet frequent kan worden veranderd (Peeters and Kroon, 2008). Daarnaast heeft de rolling stock planning een grote invloed op de planning van personeel op de voertuigen (Maróti, 2006) en spelen de kosten van de aanschaf van treinstellen een overheersende rol in de dagelijkse operationele kosten van een spoorwegmaatschappij (Cacchiani et al., 2019).

Het bovenstaande probleem kan formeel worden gezien als het *train unit assignment problem* (TUAP). In deze scriptie wordt dit TUAP bestudeerd aan de hand van het paper van Cacchiani et al. (2019). Het TUAP bepaalt het aantal aan te schaffen treinstellen, gegeven een set van geplande trips en een set van treinstellen die de spoorwegmaatschappij tot zijn beschikking heeft. De trips bevatten elk een vertrektijd, vertrekstation, aankomsttijd, aankomststation en een verwachte passagiersvraag. De beschikbare treinstellen hebben elk verschillende aanschafkosten en een capaciteit. In het bepalen van het aantal aan te schaffen treinstellen moet worden voldaan aan de verwachte passagiersvraag per trip. Daarnaast wordt een relevante extensie uitgevoerd voor dit onderzoek welke bestaat uit het analyseren van de trade-off tussen kosten en service level bij onverwachte treinuitval.

Het doel van deze scriptie is het repliceren van de methodes en resultaten van het onderzoek van Cacchiani et al. (2019) en een relevante extensie bij dit probleem beschrijven en implementeren. De contributie van deze scriptie is voornamelijk het verbeteren van de methodes uit eerder onderzoek door een uitbreiding toe te voegen aan het bestaande model, waarbij rekening wordt gehouden met het uitvallen van treinstellen. Deze scriptie focust zich op de trade-off tussen kosten en service level voor een spoorwegmaatschappij.

De structuur van deze scriptie is als volgt. Allereerst wordt een beschrijving gegeven van het probleem. Ten tweede wordt er een overzicht gegeven van de relevante literatuur voor deze scriptie. Daarna worden de methodes van het onderzoek beschreven, voor zowel de replicatie als de extensie. Vervolgens worden in hoofdstuk 6 de resultaten gepresenteerd en beschreven. Ten slotte wordt een conclusie getrokken in hoofdstuk 7.

2 Probleembeschrijving

In dit hoofdstuk worden de gebruikte parameters, restricties en doelstellingen van het TUAP probleem gespecificeerd. De set $I = \{1, ..., m\}$ bevat alle type treinstellen. Elk type $k \in I$ heeft kosten c_k per jaar, een capaciteit van R_k passagiers per treinstel en een lengte van l_k in meters. Daarnaast is de set $J = \{1, ..., n\}$ gegeven van geplande trips. Elke trip j heeft een vertrektijd van s_j, een vertrekstation, een aankomsttijd van f_j, een aankomststation en een positieve vraag naar passagiers van r_j. Elke trip wordt uitgevoerd op een lijn tussen twee stations, waarbij elke lijn een maximale treinlengte bevat. Hierdoor heeft elke trip j een maximale treinlengte van m_j in meters. De treinstellen kunnen aan elkaar worden gekoppeld om aan de passagiersvraag te voldoen mits de maximale treinlengte niet wordt overschreden, zonder restricties op de volgorde van het koppelen van verschillende treinstellen aan elkaar.

Twee trips i en j zijn *compatibel* als ze niet met elkaar overlappen in tijd en als er zich tussen de aankomst van trip i en het vertrek van trip j meer dan t minuten bevindt. In deze scriptie wordt gebruikt gemaakt van

een waarde voor t van 5 minuten. Daarnaast moet het aankomststation van trip i gelijk zijn aan het vertrekstation van trip j. De overige trips die hier niet aan voldoen zijn *incompatibel* met elkaar. Samenvattend moeten er treinstellen aan alle trips worden toegewezen dusdanig dat:

- *Dekking*: de vraag naar passagiers per trip moet volledig worden gedekt door het samenvoegen van treinstellen;
- *Maximale lengte*: voor elke trip $j \in J$, mag een trein een maximale lengte van m_j bevatten;
- *Aaneenschakeling*: twee trips kunnen na elkaar worden uitgevoerd als er wordt voldaan aan bovenstaande regel.

De doelstelling van het *train unit assignment problem* is het minimaliseren van de kosten per jaar voor het aantal aan te schaffen treinstellen.

3 Literatuurstudie

Er is aanzienlijke hoeveelheid onderzoek gedaan naar het TUAP en verdere rolling stock problemen in het treinvervoer. Het meest relevante onderzoek voor deze scriptie is het onderzoek van Cacchiani et al. (2019). Zij bestuderen het *train unit assignment problem* in een realistische toepassing van rolling stock planning voor een regionale spoorwegmaatschappij in Noord-Italië. Doordat het TUAP zelf NP-moeilijk is, maken ze gebruik van een heuristiek algoritme gebasseerd op de oplossing van een gerestricteerd probleem in een *peak period*. De auteurs concluderen dat hun heuristiek optimale of nabij-optimale oplossingen vindt welke op basis van oplossingskwaliteit en computatietijd de rest van de literatuur met vergelijkbare methodes overklast. Het onderzoek van Cacchiani et al. (2019) maakt, in tegenstelling tot dit onderzoek, gebruik van een additionele doelstelling om de deadhead verplaatsingskosten te minimaliseren. Deze kosten gaan gemoeid met het verplaatsen van treinstellen tussen stations zonder het verplaatsen van passagiers. Daarnaast legt het paper een restrictie op het maximaal aantal wagons per trein, waarbij in dit onderzoek rekening wordt gehouden met de maximale lengte van een trein.

Het grootste deel van het werk in de literatuur focust op het bepalen van een rolling stock planning waarbij aan de passagiersvraag voldaan moet worden. Het dekken van de passagiersvraag wordt in deze gevallen meegenomen als een zogenaamde *hard constraint*. Dit is ook het geval in het paper van Schrijver (1993) waarbij een circulerende treinstel verdeling wordt bepaald voor de Nederlandse Spoorwegen per dag, waarbij het rangeren en koppelen van treinstellen aan elkaar buiten beschouwing wordt gelaten.

In de meeste literatuur wordt het TUAP bestudeerd met een een klein aantal verschillende types treinstellen (twee in het meeste gevallen), zoals het geval in deze scriptie waarbij er gebruikt wordt gemaakt van drie verschillende types treinstellen. Andere onderzoeken bestuderen het geval van één treinstel type, zoals het paper van Ben-Khedher et al. (1998) waarbij de doelstelling is om de de verwachte winst van een spoorwegmaatschappij te maximaliseren. Een vergelijkbare methode is gebruikt in Abbink et al. (2004) waarbij een integer lineair programmering (ILP) model wordt gebruikt. Hierbij wordt als doelstelling het zitplaatstekort geminimaliseerd tijdens de spitsuren. Fioole et al. (2006) maken ook gebruik van een ILP model, waarbij de auteurs methodes gebruiken om de relaxatie van het probleem te verbeteren. Hierbij houden ze rekening met onderweg verbinden en splitten van treinen. Door deze methode lukt het om nabij-optimale oplossingen te genereren voor realistische instanties.

4 Methode Replicatie

Het idee van het *Peak Period Heuristiek* van Cacchiani et al. (2019) is om een set van kritieke trips te bepalen. Deze set van kritieke trips bevat trips die niet na elkaar kunnen worden uitgevoerd. Voor deze trips wordt het TUAP opgelost en bepaald hoeveel treinstellen er nodig zijn. Daarna worden alle trips een voor een aan treinstellen gekoppeld, dusdanig dat het aantal treinstellen bepaald uit de kritieke trips niet wordt overschreden. Als er geen nog te dekken trips meer over zijn, is de gevonden oplossing optimaal. Zijn er nog wel te dekken trips over, dan worden in de feasibility fase het minimaal aantal extra treinstellen aangeschaft om alle trips te dekken.

In dit hoofdstuk zal dit *Peak Period Heuristiek* worden beschreven en toegelicht. Dit heuristiek bestaat uit het *Peak Period Lower Bound* (PPLB) en de *Treinstel Toewijzing* met de fases: initialisatie, constructieve fase, feasibility fase en de iteratieve procedure.

4.1 Peak Period Lower Bound

Het doel van de *Peak Period Lower Bound* is om een zo groot mogelijke set S te vinden van onverenigbare trips aan de hand van een ongerichte gewogen graaf $G = (V, E)$. Deze graaf is gedefinieerd zodat elke knoop in V een trip j representeert met gewicht r_j. Knopen i en j zijn met elkaar verbonden als trip i en j na elkaar kunnen worden uitgevoerd met hetzelfde treinstel, zodat het aankomststation van de ene trip gelijk is aan het vertrekstation van de andere trip en er minimaal t minuten tijd tussen aankomst en vertrek zit. Om een set S van onverenigbare trips te bepalen moet er een zogenaamde *maximum-gewogen onafhankelijke set* worden bepaald in G. Een onafhankelijke set is een set van niet-aangrenzende knopen. Deze kan worden gevonden door G te transformeren in een gerichte graaf $\bar{G} = (\bar{V}, \bar{A})$ door een bron σ en een put τ toe te voegen. De bron σ wordt verbonden aan alle knopen in $\bar{G}$ zonder ingaande pijlen. Alle knopen zonder uitgaande pijlen worden verbonden aan de put τ. Met deze graaf kan het lineair programmeringsprobleem (1) - (4) worden opgelost, waarbij beslissingsvariabele x_{ij} de stroom representeert over de pijl van knoop i naar j en binaire parameter a_{ij} aangeeft of er een pijl loopt van knoop i naar j.

$$
\begin{array}{lll}
\text{Minimize} & \sum_{j \in \bar{V}} a_{\sigma j} x_{\sigma j} & (1) \\
\text{Subject to} & \sum_{i \in \bar{V}} a_{ij} x_{ij} = \sum_{i \in \bar{V}} a_{ji} x_{ji} \quad \forall j \in \bar{V} \backslash \{\sigma, \tau\} & (2) \\
& \sum_{i \in \bar{V}} a_{ij} x_{ij} \geqslant r_j \quad \forall j \in \bar{V} \backslash \{\sigma, \tau\} & (3) \\
& x_{ij} \geqslant 0 \quad \forall i, j \in \bar{V} & (4)
\end{array}
$$

Bovenstaand model minimaliseert de stroom vanuit de bron, rekeninghoudend dat de stroom in elke knoop behouden wordt en dat de stroom die een knoop ingaat minstens gelijk is aan het gewicht van de knoop. De set S kan worden bepaald door de *schaduwprijs* van restrictie (3) voor elke trip te bepalen. Bij trips met een schaduwprijs gelijk aan 1, resulteert een vraagtoename van 1 passagier tot een toename van 1 aan de doelfunctie. Trips met een schaduwprijs van 1 worden toegevoegd aan de set S, omdat deze trips de minimale vraag bepaald die een serie van treinstellen moet hebben om meerdere trips te dekken. De trips met een schaduwprijs gelijk aan 0, behoren niet tot de set S.

De werkelijke PPLB wordt berekend door de optimale toewijzing van treinstellen aan de set S te bepalen. Laat beslissingsvariabele w_j^k de hoeveelheid treinstellen representeren van type k voor trip j. De *Peak Period Lower Bound* kan berekend worden met behulp van het lineair programmeringsprobleem in de vergelijkingen (5) - (8).

$$
\begin{array}{lll}
\text{Minimize} & \sum_{j \in S} \sum_{k \in I} c_k w_j^k & (5) \\
\text{Subject to} & \sum_{k \in I} R_k w_j^k \geqslant r_j \quad \forall j \in S & (6) \\
& \sum_{k \in I} l_k w_j^k \leqslant m_j \quad \forall j \in S & (7) \\
& w_j^k \geqslant 0 \quad \forall j \in S, \forall k \in I & (8)
\end{array}
$$

Bovenstaand model minimaliseert de kosten voor de gebruikte treinstellen om de trips in de set S te dekken. Daarbij wordt rekening gehouden met de passagiersvraag per trip en de maximale lengte van van elke trip. De resulterende oplossing kan als een ondergrens worden beschouwd voor het TUAP doordat aan elke restrictie wordt voldaan voor een subset van trips die allemaal incompatibel zijn met elkaar.

4.2 Treinstel Toewijzing

In dit hoofdstuk wordt een heuristiek algoritme gepresenteerd om het TUAP op te lossen door vanuit het model (5) - (8) een oplossing te vinden voor het gehele probleem. Daarbij maken we gebruik van twee resultaten uit hoofdstuk 4.1, de set S van onverenigbare trips en de uitkomst van de beslissingsvariabele w_j^k. Hieruit kunnen we bepalen hoeveel treinstellen er per type nodig zijn voor de PPLB, zeg $\bar{w}_k$. Als er een toegelaten oplossing wordt gevonden met het heuristiek in de constructieve fase (als alle trips gedekt zijn met de alle treinstellen gebruikt in PPLB), is dit ook de optimale oplossing. De reden hiervoor is dat de kosten van deze toegelaten oplossing dan gelijk is aan de waarde van de ondergrens.
Het heuristiek algoritme wordt iteratief uitgevoerd in de volgende drie fases:

- *Initialisatie*: initialiseren van de sets van kritieke en niet-kritieke trips.
- *Constructieve fase*: berekent een mogelijke niet-toegelaten oplossing waarbij alleen de $\bar{w}_k$ treinstellen worden gebruikt. Als een toegelaten oplossing wordt gevonden, dus alle trips zijn gedekt, is dit de optimale oplossing van het TUAP.
- *Feasibility fase*: als er trips nog niet gedekt zijn, wordt er een toegelaten oplossing gevonden met extra treinstellen.

4.2.1 Initialisatie

De totale set van trips wordt gesplitst in de set *KT* van *kritieke trips* en de set *NKT* van *niet-kritieke trips*, waarbij geldt dat *NKT* $:= J \setminus$ *KT*. De set *KT* wordt geïnitialiseerd als de set van onverenigbare trips S. Elke set wordt in chronologische volgorde geordend volgens de vertrektijden van elke trip.

4.2.2 Constructieve fase

Het doel van deze fase is om een oplossing te genereren, waarbij alleen gebruikt wordt gemaakt van $\bar{w}_k$ treinstellen van elk type $k \in I$. Het idee is om per trip een set van treinstellen te bepalen die deze trip kan dekken, rekeninghoudend met de lengte en passagiersvraag van de trip. Daarna kan per treineenheid type k een *Single Depot Vehicle Scheduling Problem* (SDVSP) worden opgelost om een oplossing te genereren (Cacchiani et al., 2013). De constructieve fase is opgedeeld in twee delen: de toewijzing van de kritieke trips en van de niet-kritieke trips.

Toewijzing van Kritieke Trips

De toewijzing van de kritieke trips is uitgevoerd in vijf stappen. De pseudocode voor deze stappen is gepresenteerd in Algoritme 2 in Appendix A.1.

1. Beschouw een trip uit *KT* in de volgorde van vertrektijden.
2. Voor elke trip bepaal een subset *TU* van treinstellen die de trip kan dekken. Hierbij wordt rekening gehouden met de passagiersvraag en de capaciteiten van de treinstellen (zie *Selectie van Treinstellen Types*).
3. Los het *Single Depot Vehicle Scheduling Problem* op gegeven de huidige toewijzing en bepaal daarmee het aantal te gebruiken treinstellen per type (zie *Single Depot Vehicle Scheduling Problem Oplossing*).

4. Als voor elk geselecteerde treinstel type k het aantal te gebruiken treinstellen kleiner of gelijk is aan $\bar{w_k}$, accepteer de oplossing voor deze trip. Gebruik anders een andere trein eenheid selectie voor deze trip. Als geen selectie leidt tot een oplossing kleiner of gelijk aan $\bar{w_j}$, laat deze trip dan ongedekt (welke wordt gedekt in de feasiblity fase). Ga door naar de volgende trip en herhaal het proces tot elke trip is beschouwd.
5. Als het maximaal aantal iteraties behaald is of er zijn geen ongedekte kritieke trips meer, stop het algoritme. Anders wordt de counter i vergroot en wordt de volgorde van trips in *KT* veranderd. Voor elke nog te dekken trip $j \in$ *KT*, als j zich ook in de set S bevindt verschuif deze trip na alle andere trips in S maar voor de andere wel gedekte trips in *KT*. Een willekeurige volgorde wordt gebruikt voor de ongedekte trips.

Selectie van Treinstellen Types

Voor elke trip j bepalen we een subset van treinstellen zodat de som van de capaciteiten groter of gelijk is aan de passagiersvraag van de trip. Dit wordt gedaan zodat het verschil tussen de vraag en de capaciteit zo klein mogelijk is, om overcapaciteit te voorkomen. Als in het algoritme het aantal treinstellen geselecteerd in het *Single Depot Vehicle Scheduling Problem* groter is dan $\bar{w_k}$, moet er een nieuwe combinatie van treinstellen worden bepaald. Deze methode wordt herhaald tot een combinatie voldoet aan de kosten van AP of alle mogelijke combinaties beschouwd zijn.

In deze scriptie worden drie verschillende versies van het TUAP onderzocht. De eerste versie selecteert de treinstellen op basis van niet-afnemende volgorde in capaciteit, om overcompensatie ten opzichte van de vraag te voorkomen (*Versie Cacchiani et al.*). Daarnaast wordt de prestatie van het heuristiek beschouwd waarbij de subsets van treinstellen worden geordend op basis van kosten (*Versie Kosten*). De subsets worden in niet-afnemende waarde in kosten geordend. Naast de versies *Cacchiani et al.* en *Kosten*, wordt de prestatie van het heuristiek ook bekeken vanuit de initiële oplossing van de PPLB (*Versie PPLB*). Vanuit het Peak Period model in vergelijkingen (5) - (8) wordt voor de set van kritieke trips (S) het aantal aan te schaffen treinstellen bepaald $\bar{w_k}$. Er wordt dus afgeweken van het artikel, en het heuristiek wordt uitgevoerd door het toevoegen van niet-kritieke trips aan de al toegewezen kritieke trips, om zo een toegelaten oplossing te verkrijgen.

Single Depot Vehicle Scheduling Problem Oplossing

Met behulp van de oplossing van het *Single Depot Vehicle Scheduling Problem* (SDVSP) wordt per treineenheid type k het aantal te gebruiken treinstellen bepaald en de optimale volgorde van het gebruik van de treinstellen over de trips. Gegeven een set van trips die voor een type k kan worden gebruikt wordt een gerichte graaf $G_{SDVSP} = (V', A)$ geconstrueerd. De set V' bestaat uit de trips die op dit moment gekoppeld zijn aan treinstel k. Als trip j uitgevoerd kan worden na trip i, wordt er een pijl toegevoegd van knoop i naar knoop j met een gewicht gelijk aan nul. Aan de set van knopen wordt een bron toegevoegd waaruit pijlen vertrekken naar elke knoop in G_{SDVSP} met gewicht c_k. Daarnaast wordt vanuit elke knoop een pijl toegevoegd naar de put, met wederom een gewicht gelijk aan nul.

Met behulp van deze graaf, de binaire beslissingsvariabele y_{ij} die aangeeft of trip j uitgevoerd wordt na trip i en de binaire parameter b_{ij} die aangeeft of er een pijl loopt van i naar j kan het onderstaande *Single Depot Vehicle Scheduling Problem* worden opgelost.

Minimize	$\sum_{i \in V'} \sum_{j \in V'} c_{ij} y_{ij}$		(9)
Subject to	$\sum_{i \in V'} b_{ij} y_{ij} = 1$	$\forall j \in V'$	(10)
	$\sum_{i \in V'} b_{ji} y_{ji} = 1$	$\forall j \in V'$	(11)
	$y_{ij} \geqslant 0$	$\forall i \in V', \forall j \in V'$	(12)

Dit model in vergelijkingen (9) - (12) minimaliseert de kosten (gebruikte treinstellen) rekeninghoudend dat elke trip wordt bezocht. Doordat de restrictiematrix unimodulair van aard is, hoeft de restrictie dat y_{ij} binair

is niet worden beschouwd.

Toewijzing van Niet-Kritieke Trips
De niet-kritieke trips worden beschouwd in chronologische volgorde van vertrektijden. De toewijzing van de trips is gelijk aan de toewijzing van de kritieke trips, zonder stap 5 in *Toewijzing van Kritieke Trips*. Zoals ook geld voor de kritieke trips, worden de ongedekte niet-kritieke trips nog meegenomen in de oplossing in de feasibility fase.

4.2.3 Feasibility Fase

Als er na de constructieve fase geen ongedekte trips meer over zijn, is de optimale oplossing (die gelijk is aan de PPLB) bereikt. Is dat niet het geval wordt in deze fase de laatste ongedekte trips nog gedekt. De nog te dekken trips worden gedekt door de treinstellen van elk type door de stappen te volgen van de *Toewijzing van Niet-Kritieke Trips*. Wanneer de oplossing van het SDVSP niet voldoet aan het aantal treinstellen van de ondergrens, worden de treinstellen die te weinig zijn toegevoegd aan de huidige oplossing. Hierdoor wordt een toegelaten oplossing gecreëerd met zo min mogelijk extra aan te schaffen treinstellen.

4.2.4 Iteratieve Procedure

De constructieve en feasibility fase worden na elkaar uitgevoerd voor een aantal van n_{iter} iteraties, behalve als eerder een optimale oplossing wordt gevonden. Voor elke iteratie wordt de volgorde van kritieke trips gegeven door de vorige iteraties, waarbij zoals eerder vermeld de niet-kritieke trips altijd gebruik maken van een chronologische volgorde. In de feasibility fase wordt na elke iteratie de laagste kosten opgeslagen.

5 Methode Extensie

Spoorwegmaatschappijen leggen meer nadruk op hun operationele prestaties, door aan de ene kant druk van de reizigers en aan de andere kant deregulatie van de markt voor treinvervoer (Jespersen-Groth et al., 2009). Daarnaast blijven spoorwegmaatschappijen (grotendeels) private bedrijven met kostenminimaliserende motieven. Deze tweestrijd tussen kosten aan de ene kant en service level aan de andere kant zorgt voor een probleem. Geïnspireerd op het onderzoek van Lusby et al. (2018), wordt dit probleem bekeken onder het scenario van mogelijke treinuitval. Het onderzoek van Lusby et al. (2018) geeft namelijk een overzicht van relevante literatuur op het gebied van robuustheid in treinvervoer planning. Het beschreven probleem uit zich in de vraag: Of meer 'reserve' treinstellen aanschaffen om zo alle passagiers te kunnen vervoeren bij mogelijke uitval van treinstellen of accepteren dat niet aan alle vraag kan worden voldaan en de kosten minimaal houden?

De extensie van deze scriptie stelt een methodiek voor om inzicht te geven in deze vraag. Het doel is niet te komen tot een uiteindelijke minimale oplossing, maar de methodiek resulteert in een overzicht voor de bovenstaand beschreven trade-off in het geval dat er rekening wordt gehouden met mogelijke treinuitval waarbij spoorwegmaatschappijen zelf kunnen beslissen welke keuze zij maken in het aantal aan te schaffen *safety stock*.

5.1 Peak Period Heuristiek met Treinuitval

Het *Peak Period Heuristiek*, zoals beschreven in hoofdstuk 4.2, resulteert in een toegelaten (mogelijk optimale) oplossing voor het TUAP waarbij alle trips zijn gedekt door verschillende types treinstellen. Met deze

oplossing kan worden bepaald, door eenmalig het *Single Depot Vehicle Scheduling Problem* op te lossen, welke trips worden gedekt door elk treinstel.

Om rekening te houden met mogelijke treinuitval in het bepalen van het aantal aan te schaffen treinstellen, wordt er aangenomen dat vóór het begin van alle trips (op het begin van de dag) wordt geconstateerd dat een willekeurig treinstel kapot is en niet kan worden gebruikt. Het idee is om voor verschillende versies het probleem van begin af aan opnieuw op te lossen met behulp van de *Treinstel Toewijzing* van het *Peak Period Heuristiek*, waarbij de versies allemaal een verschillend aantal (of geen) treinstellen als 'safety stock' bevatten om trips te dekken bij treinuitval. Alle trips worden opnieuw (mogelijk op een nieuwe manier) aan treinstellen gekoppeld, voor elke versie, en zo worden de kosten van de treinstellen berekend en het tekort aan zitplaatsen. Deze procedure wordt drie keer uitgevoerd per versie safety stock, waarbij elke keer een ander type treinstel kapot gaat. Dit resulteert in drie waardes van zitplaatstekorten per safety stock. Van deze waardes wordt een gewogen gemiddelde genomen op basis van het aantal treinstellen per type in de optimale oplossing van de replicatie van *Versie Kosten*. De pseudocode van dit zogenaamde *Peak Period Heuristiek met Treinuitval* is weergegeven in Algoritme 1.

Algorithm 1 Peak Period Heuristiek met Treinuitval.

```
INITIELE OPLOSSING VERSIE KOSTEN
SAFETY STOCK: set van alle versies safety stock, elk met een verschillend aantal extra treinstellen per type
for elke safety stock do
  for elk treinstel type do
    Update aantal treinstellen per type door kapotte treinstel van huidige type te verwijderen en (mogelijke) safety stock toe te voegen
    Los Treinstel Toewijzing op
    for elk ongedekte trip do
      Bepaal het aantal te weinig passagiersstoelen
    end for
    Bereken TREINKOSTEN: kosten voor aantal aan te schaffen treinstellen
    Bereken ZITPLAATSTEKORT: aantal te weinig passagiersstoelen
  end for
end for=0
```

6 Resultaten

In dit hoofdstuk worden de resultaten van het onderzoek gepresenteerd. Ten eerste worden de instanties beschreven waarvoor het heuristiek is getest. Daarna worden de resultaten van de replicatie van de methodes uit Cacchiani et al. (2019) gepresenteerd en beschreven. Ten slotte worden de resultaten weergegeven van de beschreven extensie.

We beschouwen in totaal vier instanties om de prestaties van het heuristiek te testen. De instanties bevatten achtereenvolgens 200, 318, 668 en 1010 verschillende uit te voeren trips. Voor elke instantie zijn 3 treinstellen beschikbaar elk met verschillende kosten, capaciteit en een lengte in meters. De waardes voor deze parameters zijn weergegeven in de onderstaande tabel 1.

Tabel 1: Gegevens beschikbare drie Treinstellen.

	Kosten	Capaciteit	Lengte
Treinstel 1	230000	500	100
Treinstel 2	190000	360	75
Treinstel 3	330000	640	125

6.1 Replicatie

De resultaten van het replicatie gedeelte van deze scriptie zijn te vinden in onderstaande tabel 2. Hierin is per instantie de waarde van de *Peak Period Lower Bound* weergegeven. Daarnaast zijn de minimale kosten gepresenteerd per instantie voor alle drie de bekeken versies: *Cacchiani et al.*, *Kosten* en *PPLB*. Verder is de gap tussen de minimale oplossing en de ondergrens weergegeven in procenten. Daarnaast de computatietijd per oplossing in secondes. In de laatste kolom zijn het aantal aan te schaffen treinstellen per type treinstel weergegeven.

Tabel 2: Minimalisatie van Kosten voor TUAP voor de vier Instanties.

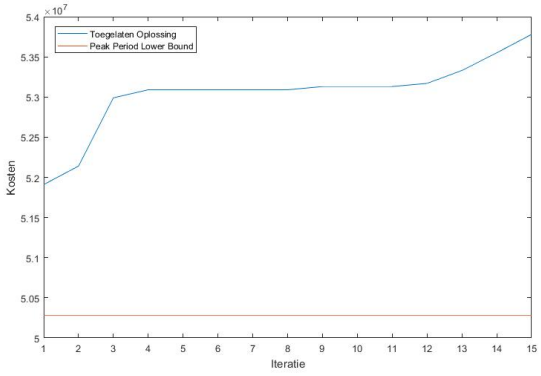
	Versie	PPLB	Kosten	Gap	Computatie-tijd	Aantal treinstellen
Instantie 1	*Cacchiani et al.*	50,280,000	57,090,000	13.54%	28	(119, 80, 44)
	Kosten		51,910,000	3.24%	31	(123, 67, 33)
	PPLB		52,680,000	4.77%	34	(121, 70, 35)
Instantie 2	*Cacchiani et al.*	52,190,000	65,240,000	25.00%	58	(132, 95, 51)
	Kosten		62,030,000	18.85%	67	(138, 83, 44)
	PPLB		62,980,000	20.67%	89	(131, 93, 46)
Instantie 3	*Cacchiani et al.*	77,590,000	99,200,000	27.85%	475	(185, 154, 83)
	Kosten		92,130,000	18.74%	532	(186, 126, 77)
	PPLB		93,020,000	19.89%	590	(175, 144, 77)
Instantie 4	*Cacchiani et al.*	125,140,000	165,870,000	32.55%	2236	(279, 273, 151)
	Kosten		159,020,000	27.07%	2481	(330, 236, 116)
	PPLB		161,290,000	28.89%	2692	(282, 270, 137)

Uit deze resultaten volgt dat de waarde van de PPLB nooit wordt behaald, wat betekent dat geen een van de drie versies een optimale oplossing creëert. Verder zijn alle vier de instanties consistent in het feit dat het heuristiek van de *Versie Kosten* het beste presteert in vergelijking met de andere twee. Dit is te zien doordat deze versie per instantie de kleinste gap met de ondergrens bezit. Opvallend is dat als het aantal uit te voeren trips groter wordt ook de gaps met de ondergrens groter worden.

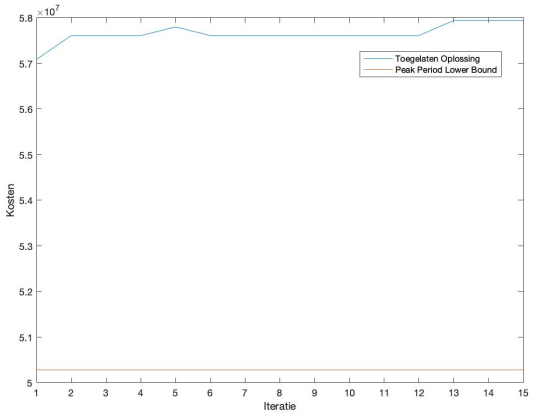
Uit verdere analyse blijkt dat de minimale oplossing in alle instanties wordt bereikt in de eerste iteratie bij de *Versie Kosten*. Dit is gepresenteerd voor instantie 1 in figuur 1a. In het heuristiek wordt na de eerste iteratie de (mogelijke) ongedekte trips toegevoegd aan de set van kritieke trips en dus in de volgende iteratie als een van de eerste ingepland (zie *Toewijzing van Kritieke Trips* in hoofdstuk 4.2.2). Deze keuze van Cacchiani et al. (2019) lijkt dus weinig tot geen invloed te hebben op het verbeteren van de oplossing. Dit resultaat wordt onderbouwd door het verloop van de toegelaten oplossing over de iteraties voor de *Versie Cacchiani et al.* zoals te zien in figuur 1b voor instantie 1. Logischerwijs is geen vergelijkbaar figuur te construeren voor *Versie PPLB*, doordat deze versie geen gebruikt maakt van een iteratief proces. In deze versie is namelijk de set van kritieke trips altijd hetzelfde (de set S van het *Peak Period Lower Bound*), waardoor een iteratief proces geen effect heeft op de oplossing.

Figuur 1: Analyse aantal Iteraties na bereiken Minimale Oplossing voor Instantie 1 Versie Kosten en Cacchiani et al.

(a) Versie Kosten.



(b) Versie Cacchiani et al.



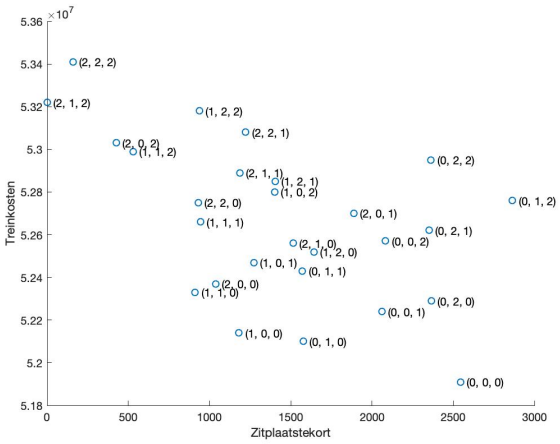
6.2 Extensie

Het beschreven *Peak Period Heuristiek met Treinuitval*, zoals weergegeven in hoofdstuk 5.1, wordt uitgevoerd aan de hand van de best presterende *Versie Kosten* voor de vier instanties beschreven in hoofdstuk 6. Voor het aantal mogelijkheden voor de hoeveelheid 'safety stock' waarvoor het heuristiek met treinuitval wordt getest, is in deze scriptie gekozen voor alle mogelijke combinaties voor de drie treinstellen types van nul tot twee extra treinstellen. Dit komt neer op een aantal van 27 versies safety stock. Daarnaast is gekozen om per iteratie precies een treinstel uit te laten vallen. Dezelfde gegevens van treinstellen zoals bij de replicatie is van toepassing op deze extensie (weergegeven in tabel 1).

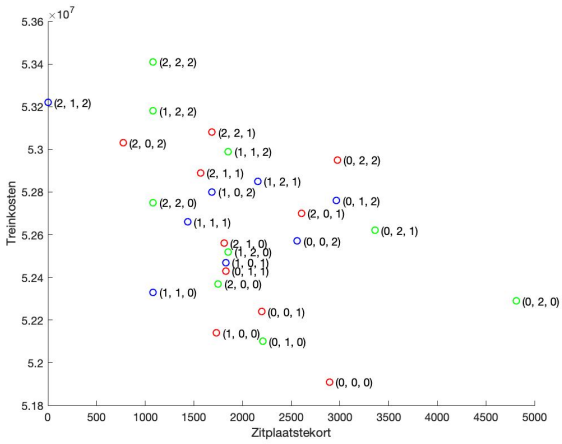
Per instantie zijn de gemiddelde gewogen waardes voor elke safety stock geplot en de worst-case scenario voor elke safety stock. Dit worst-case scenario bevat de waarde waarvan het zitplaatstekort het hoogste is, waarbij rood correspondeert met treinstel type 1, blauw met treinstel type 2 en groen met treinstel type 3. De versies in alle onderstaande figuren zijn gerepresenteerd in de vorm (x, y, z) waarbij x, y en z corresponderen respectievelijk met het aantal safety stock voor type 1, 2 en 3.

Figuur 2: Resultaten Instantie 1.

(a) Gewogen gemiddelde van drie treinstel types.



(b) Worst-case scenario.



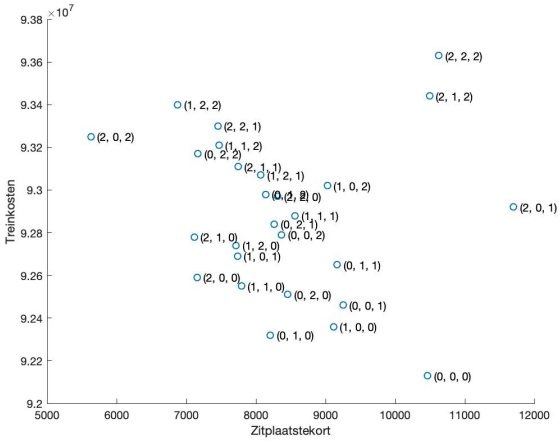
De resultaten voor instantie 1 zijn weergegeven in figuur 2. De resultaten in figuur 2a duiden op de trade-off tussen treinkosten en service level. Wanneer er minder safety stock wordt aangeschaft, zijn de treinkosten laag en het zitplaatstekort hoog. Zoals verwacht resulteert de safety stock $(0, 0, 0)$ in de laagste treinkosten en $(2, 2, 2)$ in de hoogste treinkosten. Opmerkelijk gezien resulteren deze safety stocks respectievelijk niet in het hoogste en laagste zitplaatstekort. Een verklaring hiervoor is dat het resultaat een gewogen gemiddelde van drie waardes betreft, waarbij het gebruik van meerdere datapunten per uitgevallen treinstel type een representatiever resultaat kan genereren. In figuur 2b zijn de worst-case scenario zitplaatstekorten per safety stock weergegeven. In totaal zijn 10 worst-case zitplaatstekorten van treinstel type 1, 8 van treinstel type 2 en 9 van treinstel type 3. Er is geen eenduidig treinstel type dat het slechtste presteert op basis van de zitplaatstekorten.

De resultaten voor instantie 2 zijn te vinden in figuur 4 in Appendix A.2. Zoals ook het geval was bij instantie 1, resulteert de safety stock $(2, 2, 2)$ niet in het laagste zitplaatstekort. In tegendeel tot instantie 1, resulteert safety stock $(0, 0, 0)$ wel in het hoogste zitplaatstekort. Voor instantie 2 duidt de worst-case scenario analyse ook niet op een eenduidig treinstel type dat het minst goed presteert op basis van zitplaatstekort.

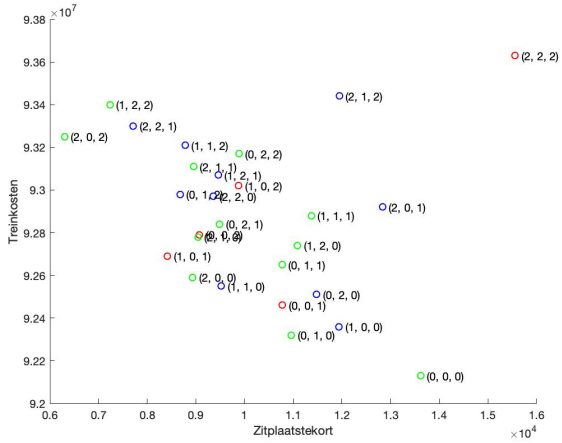
Treinstel type 1 bevat namelijk 10 worst-case zitplaatstekorten, treinstel type 2 en 3 bevatten er respectievelijk 9 en 8.

Figuur 3: Resultaten Instantie 3.

(a) Gewogen gemiddelde van drie treinstel types.



(b) Worst-case scenario.



De resultaten voor instantie 3, zoals weergegeven in figuur 3, duiden op hetzelfde resultaat van de trade-off tussen treinkosten en service level. Hierbij is de uitschieter van safety stock $(2, 2, 2)$ in figuur 3a opmerkelijk. De verklaring hiervoor is, zoals te zien in figuur 3b, dat de worst-case scenario van deze safety stock relatief een hoog zitplaatstekort oplevert. Gecombineerd met het feit dat deze worst-case scenario van treinstel type 1 is, welke zorgt voor een hoger gewicht voor het gemiddelde doordat dit type meer treinstellen bevat, kan dit resultaat worden verklaard. Wederom duiden de resultaten van de worst-case scenario analyse niet eenduidig op een slechtst presterend treinstel.

De resultaten van instantie 4, gepresenteerd in figuur 5 in Appendix A.2 duiden op gelijke resultaten als de voorgaande instanties. De trade-off tussen treinkosten en service level is duidelijk waarneembaar en geen treinstel type presteert significant het slechtst op basis van zitplaatstekort.

7 Conclusie

In deze scriptie is het *train unit assignment problem* (TUAP) bestudeerd aan de hand van het onderzoek van Cacchiani et al. (2019). Dit probleem zoekt het minimaal aantal aan te schaffen treinstellen voor een set van trips, waarbij voor elke trip wordt voldaan aan de passagiersvraag en de maximale lengte niet overschreden wordt. De trips bevatten een vertrektijd, vertrekstation, aankomsttijd en aankomststation. Om het minimaal aantal aan te schaffen treinstellen te bepalen wordt de optimale aaneenschakeling van treinstellen bepaald aan de trips. Hierbij wordt de regel gehanteerd dat twee trips na elkaar kunnen worden uitgevoerd wanneer de aankomst- en vertrekstations met elkaar corresponderen en er een aantal minuten rangeertijd tussen de trips zit.

Dit probleem is opgelost aan de hand van het *Peak Period Heuristiek* zoals gepresenteerd in Cacchiani et al. (2019). Naast de methode van dit onderzoek (*Versie Cacchiani et al.*), zijn er twee afwijkende versies van het heuristiek bestudeerd (*Versie Kosten en PPLB*). Uit de resultaten kan geconcludeerd worden dat de *Versie Kosten*, waarbij de mogelijke combinaties van treinstellen gesorteerd wordt op basis van niet-afnemende kosten, het beste presteert. Een verklaring hiervoor kan zijn, dat het heuristiek gebaseerd is op de *Peak Period Lower Bound* welke ook treinstellen selecteert aan de hand van minimale kosten. Daarnaast kan worden geconcludeerd dat, anders dan bij de resultaten van Cacchiani et al. (2019), nooit de optimale oplossing voor een van de vier instanties wordt bereikt. Een verklaring hiervoor kan zijn het verschil in gebruikte instanties. Om dit resultaat daadwerkelijk één op één met elkaar te kunnen vergelijken, zou de implementatie van het heuristiek moeten worden getest met dezelfde data. Hieruit kan dus niet worden geconcludeerd dat de implementatie van het heuristiek in deze scriptie slechter presteert dan de implementatie van Cacchiani et al. (2019).

Naast de replicatie van de methodes uit het genoemde onderzoek is in deze scriptie eveneens een relevante extensie op het probleem uitgevoerd. Geïnspireerd op het onderzoek van Lusby et al. (2018), is in deze scriptie treinuitval meegenomen in het probleem waarbij inzicht wordt gegeven in de trade-off tussen kosten voor het aanschaffen van treinstellen en service level. Aangenomen wordt dat aan het begin van de dag geconstateerd wordt dat één treinstel kapot is. Naast de al aangeschafte treinstellen, wordt voor elke versie van extra safety stock een nieuwe planning gemaakt van treinstellen aan trips. Hierna wordt bepaald (als dit het geval is) hoeveel passagiersstoelen de oplossing te weinig bevat. Dit is per versie safety stock drie keer uitgevoerd, waarbij elke keer een ander type treinstel kapot gaat. In hoofdstuk 6.2 is per instantie het gewogen gemiddelde van de zitplaatstekorten geplot tegen de treinkosten en het worst-case zitplaatstekort geplot tegen de treinkosten. Uit deze figuren kan worden geconcludeerd dat een trade-off tussen treinkosten en service level inderdaad bestaat. Wanneer de treinkosten namelijk stijgen leidt dit inderdaad tot een reductie in zitplaatstekort. Uit de worst-case analyse kan niet geconcludeerd worden dat een type treinstel significant gevoeliger is voor treinuitval. In de resultaten komt namelijk niet eenduidig één type treinstel naar voren dat de meeste worst-case scenario zitplaatstekorten bevat. Aan de hand van de analyses in deze scriptie kunnen managers van spoorwegmaatschappijen keuzes maken tussen de, in hun mening, optimale trade-off tussen treinkosten en service level voor het aantal safety stock waarbij rekening wordt gehouden met het uitvallen van treinstellen.

Het gepresenteerde onderzoek van de extensie bevat een limitatie. Op dit moment wordt er op het begin van de dag geconstateerd dat een willekeurig treinstel kapot is en niet kan worden gebruikt (zie hoofdstuk 5.1). Het *Peak Period Heuristiek met Treinuitval* creëert een volledig nieuwe toewijzing van treinstellen aan trips, waarbij de treinstellen bij observatie van een kapot treinstel dus niet op de juiste startposities staan voor de nieuwe toewijzingen. Dit leidt tot een limitatie vanuit een praktisch oogpunt.

In vervolgonderzoek zou gekeken kunnen worden naar het verder uitbreiden van het *Peak Period Heuristiek* waarbij rekening wordt gehouden met spoorbeschikbaarheid of andere scenario's waarbij het TUAP ontregeld wordt. Dit leidt tot verdere robuustheid van de methode voor het op te lossen probleem. Daarnaast

zou gekeken kunnen worden naar het aanpassen van het algoritme, waarbij treinuitval gedurende de dag (dynamisch) opgelost wordt. Ten slotte zou er gekeken kunnen worden naar een verklaring voor het feit dat de minimale oplossing voor alle instanties bij de *Versie Kosten* en *Cacchiani et al.* wordt bereikt in de eerste iteratie. Er is hier mogelijk ruimte voor verbetering van het huidige *Peak Period Heuristiek*.

Literatuur

Abbink, E., Van den Berg, B., Kroon, L., and Salomon, M. (2004). Allocation of railway rolling stock for passenger trains. *Transportation Science*, 38(1):33–41.

Ben-Khedher, N., Kintanar, J., Queille, C., and Stripling, W. (1998). Schedule optimization at sncf: From conception to day of departure. *Interfaces*, 28(1):6–23.

Cacchiani, V., Caprara, A., and Toth, P. (2013). A lagrangian heuristic for a train-unit assignment problem. *Discrete Applied Mathematics*, 161(12):1707–1718.

Cacchiani, V., Caprara, A., and Toth, P. (2019). An effective peak period heuristic for railway rolling stock planning. *Transportation Science*.

Fioole, P.-J., Kroon, L., Maróti, G., and Schrijver, A. (2006). A rolling stock circulation model for combining and splitting of passenger trains. *European Journal of Operational Research*, 174(2):1281–1297.

Jespersen-Groth, J., Potthoff, D., Clausen, J., Huisman, D., Kroon, L., Maróti, G., and Nielsen, M. N. (2009). Disruption management in passenger railway transportation. In *Robust and online large-scale optimization*, pages 399–421. Springer.

Lusby, R. M., Larsen, J., and Bull, S. (2018). A survey on robustness in railway planning. *European Journal of Operational Research*, 266(1):1–15.

Maróti, G. (2006). *Operations research models for railway rolling stock planning*. Eindhoven University of Technology Eindhoven, Netherlands.

Peeters, M. and Kroon, L. (2008). Circulation of railway rolling stock: a branch-and-price approach. *Computers & Operations Research*, 35(2):538–556.

Schrijver, A. (1993). Minimum circulation of railway stock. *CWI Quarterly*, 6(3):205–217.

A Appendix

A.1 Pseudocode algoritme Toewijzing van Kritieke Trips

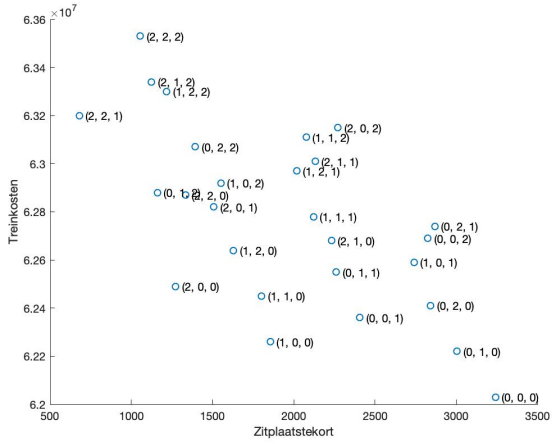
Algorithm 2 Toewijzing van Kritieke Trips.

KT: set van kritieke trips
ONGEDEKT: set van ongedekte trips
$i := 0$
while $i < n$ en *ONGEDEKT* $\neq \emptyset$ **do**
  ONGEDEKT := *KT*
  for $j \in$ *KT* **do**
    alGedekt(j) := FALSE
    FSEL(j) := set van alle subsets van treinstellen die trip j kunnen dekken
    accept := TRUE
    while FSEL(j) $\neq \emptyset$ and cover(j) = FALSE **do**
      selecteer subset $TU \in FSEL(j)$
      for $k \in TU$ **do**
        los AP(k) op
        if cost((AP(k)) > $\bar{w_k}$ **then**
          accept := FALSE
        FSEL(j) = FSEL(j) $\setminus TU$**break**
        end if
      end for
      if accept **then**
        cover(j) := TRUE
      ONGEDEKT := *ONGEDEKT* \j
      end if
    end while
  end for
  $i := i + 1$
  verander trip volgorde in *KT*
end while=0

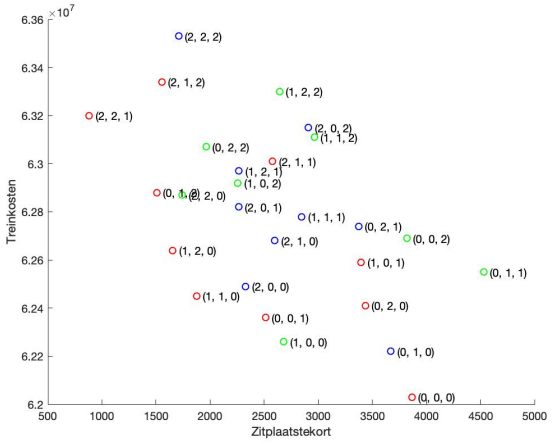
A.2 Resultaten Extensie Instantie 2 en 4

Figuur 4: Resultaten Instantie 2.

(a) Gewogen gemiddelde van drie treinstel types.

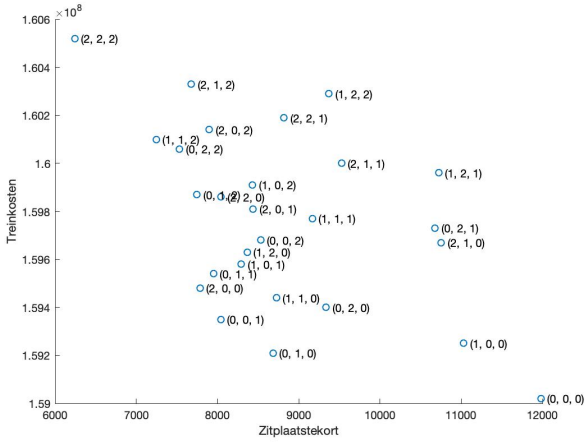


(b) Worst-case scenario.

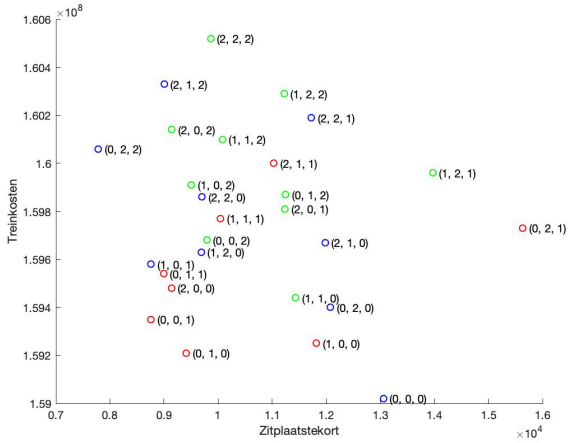


Figuur 5: Resultaten Instantie 4.

(a) Gewogen gemiddelde van drie treinstel types.



(b) Worst-case scenario.



A.3 Beschrijving Programmeercode

In dit hoofdstuk wordt een korte beschrijving gegeven van de gebruikte programmeercode die is gebruikt voor het uitvoeren van deze scriptie (zie `ProgrammeerCode_Scriptie_453777.zip`). In onderstaand overzicht wordt nauwkeurig gepresenteerd waar in de zip file de bestanden gevonden kunnen worden.

`ProgrammerCode_Scriptie_453777`

→ `Replicatie`

→ `PeakPeriodLowerBound`

`PP_LB.m`: Construeren van adjacency matrix voor PPLB model, welke in Aimms wordt opgelost.

→ `PeakPeriodHeuristiek`

`algoritme.m`: Functie voor toewijzing kritieke trips.

`algoritmeNKT.m`: Functie voor toewijzing niet-kritieke trips.

`bepaalSubsetTreinstellen_COST.m`: Bepalen van mogelijke treinstellen voor Versie Kosten.

`bepaalSubsetTreinstellen_HEURISTIEK.m`: Bepalen van mogelijke treinstellen voor Versie Cacchiani et al.

`changeKT.m`: Verander de volgorde en samenstelling van de sets KT en NKT voor de volgende iteratie.

`changeOrderKT.m`: Verander de volgorde en samenstelling van de sets KT en NKT voor de volgende iteratie.

`changeOrderNKT.m`: Verander de volgorde en samenstelling van de sets KT en NKT voor de volgende iteratie.

`cost.m`: Functie om kosten te berekenen van een oplossing.

`createAandC.m`: Construeren van A en C matrix voor het op lossen van het SDVSP in de functie solveAP.

`gappie.m`: Berekenen van de gap tussen ondergrens en oplossing.

`initialisatie.m`: Functie die de set S bepaald vanuit de schaduwprijzen.

`maakAeq.m`: Construeren van constraintmatrix voor SDVSP.

`main.m`: Main script voor het runnen van het Peak Period Algoritme.

`solveAP.m`: Functie welke het SDVSP oplost.

`toegelaten.m`: Functie welke een toegelaten oplossing construeert (Feasibility Fase).

→ `Extensie`

`algoritme.m`: Functie voor toewijzing kritieke trips.

`algoritmeNKT.m`: Functie voor toewijzing niet-kritieke trips.

`bepaalSubsetTreinstellen_COST.m`: Bepalen van mogelijke treinstellen voor Versie Kosten.

`bepaalSubsetTreinstellen_HEURISTIEK.m`: Bepalen van mogelijke treinstellen voor Versie Cacchiani et al.

`changeKT.m`: Verander de volgorde en samenstelling van de sets KT en NKT voor de volgende iteratie.

`changeOrderKT.m`: Verander de volgorde en samenstelling van de sets KT en NKT voor de volgende iteratie.

`changeOrderNKT.m`: Verander de volgorde en samenstelling van de sets KT en NKT voor de volgende iteratie.

`cost.m`: Functie om kosten te berekenen van een oplossing.

`createAandC.m`: Construeren van A en C matrix voor het op lossen van het SDVSP in de functie solveAP.

`extensie.m`: Main script voor uitvoeren extensie gedeelte.

`gappie.m`: Berekenen van de gap tussen ondergrens en oplossing.

`initialisatie.m`: Functie die de set S bepaald vanuit de schaduwprijzen.

`maakAeq.m`: Construeren van constraintmatrix voor SDVSP.

`pathbuilder.m`: Construeert paden voor treinstellen van opeenvolgende trips.

`searchNext.m`: Zoekt voor een volgende trip voor een treinstel.

`solveAP.m`: Functie welke het SDVSP oplost.

`toegelaten.m`: Functie welke een toegelaten oplossing construeert (Feasibility Fase).