Copenhagen Business School

Cand.Merc.(Mat)

Kandidatafhandling

Beslutningsstøttesystem til hægtning og parkering for DSB S-tog

Decision Support System to Assigning and Depot Planning for DSB S-tog

Forfattere
- Mathias Jensen, CMM
- Troels Nannestad, CMM

Vejleder
- Kasper Holst Hansen

Formalia
- *Antal sider: 85*
- *Antal anslag: 151.885*
- *Afl. Dato d. 22.05.2014*

_____
Mathias Jensen

_____
Troels Nannestad

Abstract

At the end of the day's operation the arriving train units should be parked at a depository while waiting to begin the tasks the following day. This problem is called the *Shunting Problem.* Planning which train units will cover the requested tasks is a complicated process and there are many factors to be included in the decision. Today this planning is conducted manually, which is a time-consuming process that requires a lot of experience and knowledge.

In this thesis we have developed a prototype of a software program that executes the shunting process efficiently, while we minimize the maintenance costs of the regular inspections, which are required for each train unit. Minimizing the costs is based on an increased focus on the restrictions associated with the maintenance inspection.

The prototype is designed as a decision support system to help out the operator to ensure a robust planning of the allocation of train units and the subsequent parking within a short computation time. We will focus on the coalition between the software and the user to achieve the best possible result.

The efficiency of the prototype is tested by small test examples, where we demonstrate how the prototype presents different scenarios, and how the operator may affect the solution afterward if he finds it possible to optimize.

At the end of this thesis we will demonstrate why our prototype is profitable and how it can ensure a reduction in overall maintenance costs by reducing the number of inspections on an annual basis. The interaction between the decision support system and the operator must ensure that the maintenance of the train units are conducted in a timely manner, to reach, but not exceed the amount of kilometers driven in correlation to the demands that DSB S-tog are assigned to.

Indholdsfortegnelse

Terminologi

Dette afsnit indeholder en definition af de vigtigste fagtermer, der bruges gennem hele opgaven.

1. Et **togsæt** er den fysiske enhed, der varetager et togløb
2. En **togkomposition** er en sammensætning af flere *togsæt*
3. Et **togløb** beskriver en opgave som et togsæt eller togkomposition tildeles.
4. Et **tognummer** beskriver delelementer i et togløb
5. En **hægtning** beskriver tildelingen mellem *togsæt/togkompositioner* og *togløb/tognumre*.
6. **Indtægtsgivende toge** omhandler toge, der udelukkende bliver brugt til transport af passagerer
7. **Ikke indtægtsgivende toge** omhandler toge, der ikke medtager passagerer, og udelukkende positionerer sig til næste opgave.
8. **Positionering** er den proces, hvor et togsæt transporteres hen til en kommende opgave for at efterkomme senere efterspørgsel. Dette kan fx være i form af et indtægtsgivende tog, eller for at sikre teknisk vedligehold senere i tidsplanen.
9. Et **depot** er den komplette infrastruktur på en station, hvor der forekommer tograngering samt parkering af togenheder. Alle depoter har faciliteter til rengøring af togsæt, og enkelte har faciliteter til vedligeholdelse.
10. Et **vedligeholdelsesdepot** er et værksted, hvor der kan udføres vedligeholdelse
11. Et **depotspor** er et spor, hvor et togsæt kan parkeres, hvis det ikke har en opgave, der skal udføres.
12. **Køremændene**, KMD, er personalet der udøver tograngering.
13. En **toglinje** indikerer en samling af ensartede togstrækninger i grupper ud fra den tid på dagen, de kører, de stationer de standser ved osv. De er ofte angivet med separate bogstaver og forskellige farver.
14. **BSS:** beslutningsstøttesystem.
15. **TUSP:** Train Unit Shunting Problem

1. Introduktion

1.1 Indledning

I hovedstadsområdet leverer DSB S-tog transport til ca. 345.000 mennesker på en almindelig hverdag. Dette kræver en nøje produktionsplanlægning, som gennem de seneste år er blevet sat i fokus, hvorfor DSB S-tog bevidst har indført nye metoder, principper og IT-systemer for at effektivisere planlægningen.

DSB S-tog har i dag 135 togsæt til at dække den daglige kørsel. Togsættene dækker opgaver hen over dagen og parkeres om natten på materieldepoter, som er placeret på enkelte udvalgte S-togsstationer. De fleste af disse stationer har begrænset depotplads bortset fra Københavns Hovedbanegård (*KH*) samt depotet i Høje Taastrup (*HTÅ*). På sidstnævnte depot ligger værkstedet, hvor der er vedligeholdelsesfaciliteter, mens visse mindre serviceopgaver kan varetages på Hundige Station (*UND*).

Hver dag ligger der en tidskrævende planlægningsopgave af både parkeringen af togsæt samt hægtningen til morgendagens togløb. Udfordringen ved parkeringen er at sikre, at udrulningen af køreplanen kan foregå uden krævende omrokeringer på depotområderne. Sådanne krydsninger er tidskrævende og er derfor ikke tilladt. Udfordringen for hægtningen er, at de togsæt, der bliver tildelt bestemte togløb i netværket, skal dække efterspørgslen samtidig med, at alle togsæt regelmæssigt skal til vedligeholdelsestjek på værkstederne. Dette sker primært ud fra en omdirigering fra KH, hvilket betyder, at det bliver planlæggerens rolle at sørge for, at alle togsæt regelmæssigt runder en overnatning på KH. Ved den kommende dags hægtning kan planlæggeren således få sendt ønskede enheder til vedligeholdelse.

Der er sat faste restriktioner for, hvornår et togsæt skal til eftersyn, og dette indtræffer senest efter 50.000 km. Ligeledes skal der foretages mindre tjek med opfyldning af bremsesand hver 10.000 km, og yderligere kan der opstå en række uforudsete fejl, hvorved skaderne også skal udbedres på værkstederne. Rammerne for omdirigering ved KH lader sig begrænse af, at der dagligt parkeres omkring 25 togsæt over natten, mens der kun er ca. 7 togløb, som afgår mod værkstederne i UND og HTÅ.

Omkostningerne forbundet med et vedligeholdelseseftersyn er høje, og DSB S-tog har derfor et øget fokus på at udnytte flest mulige kilometer, inden enhederne sendes til eftersyn.

Dette er grundlaget for de spørgsmål, som dette projekt vil forsøge at besvare.

1.2 Problemformulering

Formålet med dette projekt er at udvikle en prototype, der sikrer en effektiv planlægning af hægtning mellem togsæt og togløb samt parkeringen over natten på Københavns Hovedbanegård.

Med udgangspunkt i litterære artikler, vil vi formulere en matematisk model, der kan støtte den nuværende manuelle beslutning proces. Vi vil vise, hvordan koalitionen mellem automatisk genereret planlægning og brugeren kan sikre en effektiviseret køreplanlægning.

Til sidst vil vi undersøge, hvorvidt denne løsning kan bidrage til at nedbringe dele af de økonomiske driftsomkostninger, som er forbundet med vedligeholdelse af de enkelte togsæt.

1.3 Problemafgrænsning

Dette afsnit har til formål at beskrive de nødvendige afgrænsninger, vi har måtte tage i forhold til vores modeller og opgaven generelt. Dette har vi set som en nødvendighed for at bibeholde vores primære formål, at opbygge en prototype af et program, som skal være et hjælpeværktøj for planlægning i DSB S-tog.

Planlægningen af togdrift og udrulning af fysiske køreplaner ses bredt omdiskuteret i togindustrien, hvilket afspejles i afsnit 2.1. Her ser vi nærmere på 6 artikler, som beskriver forskellige tilgange til planlægning og løsning af togdriften, og heriblandt vælges vores teoretiske fremgangsmåde for prototypen.

Vi har, i samspil med DSB S-tog, haft et ønske om at udvikle et beslutningsstøttesystem (*BSS*), som prioriterer mulige løsningsforslag højere end globale optimale løsninger. Herved har vi valgt at begrænse vores teoretiske tilgang til at være en to-trins løsning, hvor det primære mål er at finde en lokal løsning for efterfølgende at inddrage planlæggeren. Han skal med bedste intuition forsøge at optimere det fundne valg. Begrundelsen for denne tilgang gennemgår vi i afsnit 2.2

Planlægningsfasen for DSB S-tog starter lang tid før den reelle drift afvikles, og vores BSS skal ses som et bidrag til dag-til-dag planlægningen af driften. Det betyder, at vi udelukkende fokuserer på den

kortsigtede planlægning i forhold til udviklingen af programdelen. Den kortsigtede planlægning beskrives nærmere i afsnit 3, hvor vi også belyser de elementer fra den langsigtede planlægning, som har en indvirkning på den daglige drift.

Vi har desuden valgt, at målet for vores BSS er at effektivisere og automatisere de nuværende beslutningsprocesser, som foregår ved planlægningen af driften, når der ikke optræder forstyrrelser. Ved dagens start er det derfor kendt information for planlæggeren hvilke togsæt, der vil ankomme på KH til parkering om aftenen.

I løbet af dagen er der yderligere forskellige scenarier at iagttage, eftersom der løbende sker en tilpasning til efterspørgslen. DSB S-tog er nødsaget til at tilpasse udbuddet af sæder i løbet af dagen for at imødekomme den øgede efterspørgsel, der er i myldretidstrafikken. Herved sker der løbende på- og afhægtninger af togsæt. Vi har valgt at se på den hægtning, som finder sted ved dagens start, og afgrænser os således fra den løbende skalering. Der skelnes således mellem realtidsplanlægning og dag-til-dag planlægning, hvor vi som sagt varetager den sidste.

I takt med ovenstående sker der naturligvis også parkeringer i realtiden, som følge af den dagligt varierende efterspørgsel, men vi begrænser os til at se på den planlagte parkering natten over på depoterne. Vi søger derfor en brugbar parkering, der sikrer morgenstundens drift af hægtningen. Yderligere har vi valgt at begrænse os i det infrastrukturelle netværk, så vores problemfelt kun iagttager situationen på KH.

Til sidste afgrænser vi os til at se på planlægningen af indkomne togsæt mandag – torsdag, hvor vi finder en hægtning og en parkering, som sikrer en glidende afgang af togsæt fra KH tirsdag – fredag morgen. Denne afgrænsning skyldes, at der er ens karakteristika i hverdagens drift, hvorved vi kan gruppere disse under samme scenarie.

At vi afgrænser os til udelukkende at analysere forholdene på KH og driften i hverdagen skyldes, at det ifølge DSB S-tog er på disse dage, hvor de største problematikker for planlægningen forekommer. Det er i disse dage, der er størst pres på netværket, og det er på KH, der skal foretages flest parkeringer om aftenen. Prototypen udvikles derfor, så den let kan videreudvikles til at kunne varetage både resten af netværket samt de resterende dagstyper. Bl.a. forsøger vi at give et kvalificeret bud i afsnit 6.2 på en videreudvikling af modellerne.

1.4 Metode

Vi har nu afgrænset vores problemfelt tilstrækkeligt, så vi dermed kan redegøre for valget af opgavetype og metode for det tilbageværende problem. Derudover skal afsnittet benyttes til at give læseren en forståelse for opbygningen af opgaven, samt hvilke formål, indgangsvinkler og interessenter opgaven ligger til grund for.

Metodevalg

Opgavens metodevalg er udelukkende baseret på en kvantitativ tilgang med matematisk optimering af allerede bestående arbejdsopgaver indenfor DSB S-tog. I dag bliver disse opgaver udført manuelt på baggrund af empiri og kognition og derved uden maskinel software kraft. Dette medfører, at vi har valgt en pragmatisk tilgang til løsning af vores problemformulering specielt i forhold til et BSS.

DSB S-tog – et casestudie

Vi har valgt at bearbejde vores problemfelt som et casestudie, hvor vi vil undersøge og analysere situationen for DSB S-tog i dets virkelige omgivelser. Som Robson [R02] skriver:

> *Casestudiet er en strategi til empirisk udforskning af et udvalgt nutidigt fænomen i sit naturlige sammenhæng ved anvendelse af forskellige datakilder, der kan anvendes i en bevisførelse*

[R02] beskriver primært casestudier, som oplagte værktøjer til at være teoriskabende, men samtidigt afviser han ikke, at casestudier kan være til brug for bevisførelse af givende teorier. Vores udgangspunkt er at påvise, at gode beslutningsstøttesystemer kan opbygges vha. løsningstilgangen separation og inddrages til optimering af S-togsdriften i DSB S-tog.

Ved casestudiet vælger vi at tage udgangspunkt i et konkret eksempel fra virkeligheden, hvorved vi kan anvende kendt og beskrevet teori. På denne måde kan vi opstille modeller ud fra de teoretiske forudsætninger, som kan beskrive virkeligheden i praksis.

Empiri, data og erkendelsesprocessen

Det empiriske data, vi har fået tilsendt af DSB S-tog ligger til grund for udarbejdelsen af vores opgave, hvilket betyder at vores testdata er DSB S-togs observerede data fra et dagsforløb onsdag d. 26. feb. 2014. Hertil har vi forsøgt at skabe næste dags hægtning samt den parkering, som finder sted natten over. Resultatet af den originale hægtning og deponeringsplan er derfor allerede kendt, hvormed vi kan lave en direkte sammenligning med det manuelle udførte arbejde og beslutningsstøttesystemets

forslag til selvsamme problem. Således kan vi i sidste del af opgaven fremsætte et godt sammenligningsgrundlag i forhold til fordele og ulemper, som den maskinelle supportkraft kan bidrage med.

Vi har opnået en bredere forståelse af det generelle hægtnings- og parkeringsproblem som under samlet betegnelse kaldes Train Unit Shunting Problem (*TUSP*). Dette er gjort gennem både den litterære empiri og den tilhørende operationsanalytiske teori. Den konkrete viden omkring TUSP hos DSB S-tog er sket på baggrund af kvalitative møder/interviews med deres backoffice, samt møde/interview med den operationelle part, der sidder i Materialestyrelsen (MAS) og dagligt håndterer planlægningen.

Formål

Opgavens primære formål er at lave et beslutningsstøttesystem til DSB S-tog, som kan bidrage til deres daglige planlægning for hægtningen af indkomne togsæt til udgående togløb den efterfølgende morgen. Ydermere skal systemet fremstille en tilstrækkelig parkering til natten, som sikrer en flydende udrulning af driften. Dette er gjort på baggrund af en interesse fra DSB S-togs side af, da processen i dag håndteres manuelt og kognitivt af en operatør.

Vi vil tage udgangspunkt i en række litterære artikler, som beskriver flere teoretiske tilgange til at håndtere kompleksiteten i TUSP. Vi vil dernæst give vores syn på sagen og belyse hvilke overvejelser, vi har gjort os i valget af vores tilgang.

Perspektiv

Vi har haft to overordnede indgangsvinkler til opgaven, hvilket allerede kort er blevet indikeret. Vi tager udgangspunkt i et økonomisk perspektiv, som handler om at sænke DSB S-togs driftsmæssige udgifter ved dag-til-dag planlægningen, samt et matematisk perspektiv, hvor vi forsøger at benytte tillært operationsanalytisk og matematisk teori til at skabe et støttesystem. Heri har vi inddraget software værktøjet SAS, som et hjælpeprodukt til løsningen af problemfeltet. Heri ligger også en underforståelse af, hvilke interessenter, der kunne have interesse i opgaven.

Økonomisk

Den daglige udrulning af køreplanen hos DSB S-tog er i dag udelukkende udformet af kompetente og erfarne medarbejdere, der igennem en længere årrække har været en del af den daglige kørselsdrift af S-togene. Det betyder, at de medarbejdere, der i dag sidder med planlægningsprocessen, gennem tiden har oparbejdet en række kognitive redskaber til at planlægge en tilstrækkelig god og robust

togdrift ud fra de menneskelige forudsætninger, der ligger til grund herfor. Hele omdrejningspunktet for DSB S-tog er at sikre en robust drift, samt at sikre rettidig serviceeftersyn til togsættene, således at der ikke opstår situationer, hvor den nuværende køreplan ikke længere kan overholdes ud fra de restriktioner, der måtte foreligge.

Der tegner sig hurtigt et billede af, at der i tilfælde af manglende rettidig omdirigering af togsæt vil opstå både direkte og indirekte økonomiske konsekvenser hos DSB S-tog. Direkte kan det måles ved, at togsættet ikke længere er i stand til at varetage sin kørsel, hvor der skal benyttes "flyttevogne" til at køre de givne enheder til service udenom den almindelige togdrift. Dette medfører ekstra transportomkostninger, mandskabsplanlægning etc. Indirekte påvirker det også en lang række faktorer, som er økonomisk belastende – mest af alt over tid. Her har vi set, at man eksempelvis er nødt til at stoppe driften af et togsæt på en linje, der er i brug, og omdirigere kunderne til anden materiel, der i stedet kan varetage driften. Resultaterne af dette, påpeger DSB S-tog selv, er ringe kundeoplevelser og kritik af togdriften, hvilket kan ende ud i en faldende efterspørgsel. Både som supplement til de direkte og indirekte udgifter vil aspektet i at fremstille en genopretningsplan, og den efterfølgende koordinering af denne, have en økonomisk belastning for DSB S-tog. Derfor er det ekstremt vigtigt at inddrage sikkerhedssystemer, som sikrer, at der ikke sker kognitive beslutningsfejl, hvilket kan begrænses igennem software støttesystemer. Omvendt er det heller ikke økonomisk ansvarligt for DSB S-tog, at sende et togsæt for tidligt til vedligeholdelse. Et eftersyn er omkostningsfyldt, og jo før et materiel sendes til service, desto flere eftersyn vil det opnå i sin levetid, hvilket betyder, at der benyttes unødvendige omkostninger til service.

I dag sker denne sikring ud fra princippet "better safe than sorry", hvilket viser vigtigheden af ikke at komme i ovenstående situationer. Derfor sender man så vidt som muligt togsættene til vedligeholdelse i god tid, hvilket øger antallet af eftersyn. Vi ser lidt nærmere på disse situationer i afsnit 6.3.

Vores indgangsvinkel har derfor været at udvikle et system, der kan sikre at der rettidigt, hverken for tidligt eller for sent, sendes togsæt til vedligeholdelse, således at driften ikke bliver forstyrret. Ydermere kan udviklingen af et støttesystem, der parkerer togsæt på depotsporene samt foretager hægtningen, reducere tidsforbruget til det udførende planlægningsarbejde.

Matematisk

Vi ønsker at varetage det overordnede hægtningsproblem som et binært "Lineare Sum Assignment Problem" (*LSAP*). Dette gøres ved hjælp af en "Mixed Integer Linear Problem" (*MILP*) solver i SAS, som kan varetage binære variable samt den kompleksitet, der udspringer i et større problem med de restriktioner, der måtte følge.

Vi ønsker ligeledes at udvikle et tilsvarende LSAP vha. MILP omkring parkeringsproblem, som afhænger af det dataoutput, vi genererer i hægtningsproblemet. Desuden ønsker DSB S-tog et program, der tillader en justering af den fremlagte løsning. På den måde kan den kognitive viden, som operatøren ligger inde med være behjælpelig til at afværge bl.a. flaskehalssituationer.

Restriktionerne for hægtningen ligger fastlåst, eftersom de er dikteret af transportministeriet i forhold til service og sikkerhedskrav. Derimod er der nogle forretningsregler i DSB S-tog, som foreskriver, hvornår en parkering er mulig, hvilket vi vil forsøge at teste på for at undersøge, hvornår chancerne for en brugbar løsning er størst. Dette vender vi tilbage til i afsnit 5.1.

Teoretiske forudsætninger

Det antages, at læser har en vis form for matematisk forståelse samt grundlæggende viden omkring lineær algebra og heltalsprogrammering. Desuden forventes en generel forståelse indenfor det operationsanalytiske felt, heriblandt løsning af assignmentproblemer, heltalsalgoritmer, Branch and Bound strategier mm..

1.5 Struktur

Foruden dette afsnit, som gerne på nuværende tidspunkt skulle have givet et overordnet indblik i rammerne for denne opgave, er dette underafsnit skabt for at give læseren et hurtigt overblik over de indholdsmæssige afsnit.

Afsnit 2 omhandler overordnede set, hvilke teoretiske forudsætninger og overvejelser vi har gjort os for at imødekomme vores problemformulering. Her vil vi gennemgå den litterære teori og påpege flere metodiske tilgange med deres fordele og ulemper for herigennem at begrunde vores tilvalg. Dertil vil vi beskrive, hvilke elementer vi synes, der er vigtigst at inddrage i et beslutningsstøttesystem. Til sidst vil vi præsentere det valgte software, der benyttes til at skabe rammerne omkring løsningen.

Afsnit 3 er en gennemgang af DSB S-tog og deres planlægningsprocesser. Vi ønsker her at give et overordnet billede af de rammer, som ligger til grund for at automatisere og generalisere den nuværende planlægning, således at den kan modelleres.

Afsnit 4 beskriver de to opstillede modeller, der løser hhv. hægtningen og parkeringen. Der vil her blive redegjort for de konkrete restriktioner, som gør sig gældende, samt hvilke kriterier vi ønsker at optimere i hver proces af to-trins løsningen.

Afsnit 5 bygges om op omkring en forbedring af programmet vha. tests. Her forsøger vi at varetage nogle af de fravalg, som vi har taget i forhold til andre teoretiske løsningstilgange. Dette resulterer i en udvidelse af parkeringsproblemet, som vi vælger at arbejde videre med som en del af beslutningsstøttesystemet.

Afsnit 6 og afsnit 7 afrunder og opsamler arbejdet igennem projektet. Her vil vi belyse, hvordan programmet kan forbedres og videreudvikles. Dette ender ud i en opsamlende konklusion, som kort opridser de vigtigste elementer igennem opgavens forløb.

2. Løsningsværktøj

2.1 Litterær gennemgang

Gennem det seneste årti er der blevet publiceret en stor mængde videnskabelige artikler omkring brugen af operationsanalytiske teknikker i togindustrien. Blandt andet findes der en del forskning og analyse af det hollandske tognetværk. Vi ser bl.a. Freling et al. [FLKH02], Lentink et al. [LFKW03], Schrijver et al. [SLK05], Huisman [HKLV05] og Wezel et al. [WJ08] varetage elementer i den hollandske togtrafik. Som resultat af et tæt samarbejde mellem Reiziger, der er Hollands hovedaktør for togindustrien, og de hollandske universiteter, har Holland derfor bidraget stort til den litterære udvikling på dette område. Samtidig har Holland ifølge [LFKW03] et af de hårdest belastede tognetværk, hvilket gør deres løsningsforslag og tilgange yderligere interessante.

På baggrund af seks artikler vil vi beskrive de modelmæssige overvejelser, der ligger til grund for vores løsningsforslag til DSB S-tog. Med udgangspunkt i en samlet forståelse af beslutningsprocessen gives først en redegørelse for hver enkelt artikel og efterfølgende håndteringen af nøglebegreberne, som vi finder interessante.

2.1.1 Shunting of Passenger Train Units in a Railway Station

Freling et al. [FLKH02] præsenterer det klassiske Train Unit Shunting Problem (TUSP), hvor togsæt bliver hægtet på togløb, og hvor de ser på den efterfølgende parkering. Dette munder ud i en to-trins løsning:

- Trin 1: Hægtning mellem indkommende og udgående togsæt
- Trin 2: Parkering af togsæt

Forfatterne beskriver en model for løsningen af hægtningen mellem indkommende og udgående togsæt, der via en standard MIP solver i cplex kan løse problemet efficient. Denne del kaldes også et Train Unit Problem (TMP). Anden del består af parkeringsproblemet, som kaldes Train Assignment Problem (TAP). Problemet er formuleret som et partitioneringsproblem[1] med sideløbende bibetingelser. De definerer en hægtning som mulig, såfremt der ikke forekommer krydsninger og at

[1] Partitioneringsproblem er den opgave, hvor det besluttes om et givent set S af positive heltal kan opdeles i to undergrupper S1 og S2, således at summen af tallene i S1 er lig summen af tallene i S2.

den øvre grænse på længden af sporet ikke overskrides af længden af de togsæt, der er parkeret på sporet. En stor fordel ved modellen er, at begrænsningerne med respekt til en mulig hægtning bliver betragtet implicit. Samtidig optræder der en ulempe ved, at antallet af potentielle parkeringer kan stige eksponentielt i takt med antallet af togsæt. Da antallet af hægtninger stiger eksponentielt med antallet af togsæt, vælger forfatterne at tilgå problemet med en såkaldt "*column generator*"[2], hvor en dynamisk programmeringsalgoritme benyttes til at finde mulige parkeringer på depoterne. Denne dynamiske programmering er baseret på en ny underliggende netværksstruktur. Denne netværksstruktur er videre afhængig af, hvorvidt der er frie depoter eller LIFO depoter[3] samt antallet af togsæt, der skal deponeres.

Løsningsmetoden er blevet testet på stationen Zwolle, da der forekommer mange hægtnings- og parkeringsprocesser samtidig med at kapaciteten er begrænset. Der opstilles forskellige scenarier, hvor objektfunktionen varieres i TMP, mens der i TAP ændres fra frie spor til LIFO. Der testes også på de forskellige ugedage. Resultaterne viser, at hægtningen i trin 1 tager et par sekunder, mens parkeringen i trin 2 tager et sted mellem 20-60 minutter. I størstedelen af kørslerne formår programmet at finde en mulig løsning, hvor alle enheder bliver parkeret, dog ses det, at løsningen ved LIFO frem for frie spor giver et forringet resultat.

Artiklen viser således en opdelt to-trins løsning, hvor problemfeltet ikke anskues globalt. Herved er der ingen forsikring om, at en løsning findes på trods af, at den er eksisterende. Desuden påviser de til at starte med, at deres problemfelt er NP-komplet, hvilket betyder, at beregningstiden er eksponentielt stigende. Vi vælger at benytte opsplitningen i en to-trins løsning, da det medvirker til at opdele problemet i mindre delkomponenter, som gør det lettere og hurtigere at løse.

Da Holland opererer med flere typeenheder og en større mængde af togkompositioner end DSB S-tog, vælger de i artiklen at tage udgangspunkt i at minimere på split. Dette element ønsker vi ikke at benytte os af.

[2] Den overordnede idé er, at mange lineære problemer er for store til at betragte alle variable eksplicit. Da de fleste variable vil være nonbasic og påtage sig en værdi på nul i den optimale løsning, er kun en delmængde af variable nødvendigt at overveje i teorien, når problemet skal løses. En "*column generator*" udnytter denne idé til kun at generere de variable, der har potentialet til at forbedre objekt funktion.

[3] fri depotstruktur: kan tilgås fra begge sider; LIFO (Last In First Out) eller lukket depot: kan kun tilgås fra en side af. Se yderligere beskrivelse i afsnit 4.2.2

Et kritisk punkt omkring deres model er, at den ikke varetager grundlæggende krav til de enkelt togsæt heriblandt vedligeholdelseseftersyn. Dette er en betydelig faktor, da materiellet skal overholde visse restriktioner for at kunne indgå i driften. Det vil betyde, at denne overvågning skal forekomme manuelt, og derfor ikke er en del af den samlede løsning.

2.1.2 Applying Operation Research techniques of train shunting

Lentink et al. [LFKW03] beskriver deres forskning inden for anvendelse af operationel forskningsteknik af shunting problemet. Forfatterne præsenterer TUSP, som et problem, der skal løses i et dekomponeret setup, hvor de udvikler en fire-trins algoritme, som muliggør ændringsforslag for operatøren mellem de enkelt trin. De fire trin er:

1. Hægtning mellem indgående og udgående togsæt
2. Estimering af ruteomkostninger for togsæt
3. Parkering af togsæt på depotspor
4. Ruteplanlægning for togsæt

Når operatøren skal køre programmet, så kan han i trin 1 fiksere prioriteringen af hægtningen og yderligere ændre løsningen, når programmet er færdigkørt. Når hægtningen mellem indgående og udgående togsæt er fastlagt, kan trin 2 estimere minimumsomkostningerne, som er en del af input i trin 3, hvor selve parkeringen forekommer. Herefter forsøger trin 4 at finde en mulig rute til og fra depotet for de enkelte togsæt.

Forfatternes fokus i artiklen omhandler trin 2 og trin 4, og de tager herfra udgangspunkt i en ny tilgang i beskrivelsen af infrastrukturen på en station. Denne fremstilling synliggør mulige rutekonflikter, som ellers ikke ville være opdaget. Ved brug af deres model af infrastrukturen beskriver de en søgealgoritme til at finde den optimale rute for togsættene. Algoritmen kaldes Occupied Network A* Search og indeholder visse stopkriterier, som sikrer, at algoritmen kan benyttes, selvom dele af netværket er optaget eller ikke er tilgængelig på det pågældende tidspunkt. Søgningen efter mulige ruter foregår sekventielt, da det er enormt tidskrævende at søge simultant i

praksis. Ulempen er imidlertid, at der ikke garanteres en optimal løsning ved sekventiel udvælgelse. I et forsøg på at reducere denne tilbagegang implementeres en 2-opt forbedringsstrategi[4].

Løsningsforslaget testes på stationerne Enschede og Zwolle i Holland, hvor trin-1 og trin-2 løses indenfor et minut, mens der forekommer stor beregningstid for parkeringsproblemet. Trin-4 viser en forbedret løsning ved implementeringen af 2-opt forbedringsstrategien.

Denne artikel tager udgangspunkt i samme løsningsmodel som [FLKH02], men hvor modellen bliver udvidet og den menneskelige operatør inddrages. Den samlede løsning er udvidet med yderligere to trin, trin 2 og trin 4, hvor der som udgangspunkt forsøges at varetage flere elementer i shunting processen. På trods af at løsningen stadig findes sekventielt, er fordelen ved at inddrage yderligere processer, at vi kommer tættere på et globalt optimum. På den anden side medfører yderligere restriktioner en øget kompleksitet samt en forringelse af beregningstiden. Dette kan ligeledes resultere i, at chancen for en brugbar løsning svækkes.

Overordnede set ønsker forfatterne at minimere på antallet af split, men forskellen er i høj grad, at man forsøger at inddrage operatøren, således at løsningen kan blive påvirket undervejs. Dette medfører, at man tidligt i processen kan påvirke eller tilrette de maskinelle beslutninger, som har svært ved at håndtere situationer, der kræver særbehandling i planlægningen.

Vi kan ud fra artiklen se, at ved at udvide scenariet øger vi blot løsningskompleksiteten, hvilket vi ikke ser fordelagtigt og ønsker derfor at bibeholde en to-trins opbygning, hvor hægtning og parkering er de primære elementer.

2.1.3 Shunting of Passenger Train Units: An Integrated Approach

Schrijver et al. [SLK05] beskriver en integreret metode til at løse TUSP. Forfatterne opsætter en model, der har en integreret løsning for hhv. hægtningsproblemet og parkeringsproblemet. Modellen bliver udvidet på forskellige måder i et forsøg på at inkorporere detaljer fra den virkelige verden. Den basale model, model 1, indeholder kun LIFO spor og tager udgangspunkt i et konkret eksempel på stationen Zwolle med over 300.000 bibetingelser. Udfaldet herfra bliver model 2, hvor de tilføjer

[4] Den 2-opt forbedringsstrategi antager, at de anmodet ruter er sorteret, typisk efter start tidspunkt. Strategien gennemløber nu alle ruterne. Herefter forekommer der endnu et gennemløb for hver anmodet rute for at vælge en rute med et højere indeks og for at forsøger at bytte om på rækkefølgen af disse to anmodninger.

forbedringer til model 1 ved en sammenlægning af restriktionen ved krydsninger[5] og en sammenlægning af andre betingelser i en såkaldt clique[6].

Efterfølgende opstiller forfatterne model 3, hvor de indfører restriktioner mod antallet af blandede depotspor, dvs. spor, der indeholder forskellige typer af togsæt. Dette vil gøre modellen mere robust, da rangeringen af samme type togsæt på ikke blandede spor bliver irrelevant. Til sidst opstiller de en ny model med frie spor, men da den ikke har en mulig løsning i et virkeligt setup pga. en ekstrem forøgelse af beslutningsvariable forkastes denne. Forfatterne forsøger i stedet at modellere den indgående og udgående side af sporet som en beslutningsvariabel, men også her forekommer der problematik med krydsning.

Vi ser i denne artikel, at [SLK05] forsøger at finde et globalt optimum ved en simultan løsningsalgoritme. Hægtningen og parkeringen er hermed integreret i én model, hvor antallet af opsplit samt antallet af spor, der indeholder forskellige typer af togenheder, minimeres. Dette gøres med henblik på at lette processen for udrulning af køreplanen den efterfølgende dag. Denne globale tilgang resulterer i, at forfatterne må afvise hypotesen, eftersom løsningen er for svær at finde.

Her er det interessant at se på, hvorvidt det har størst værdi at finde et globalt optimum i enkelte tilfælde, eller finde en effektivisering af den nuværende situation hver gang. Vi har til forskel valgt at vores BSS skal være løsningsorienteret, og at selv små processer og forbedringer kan være fordelagtige for den menneskelige operatør.

En yderligere ulempe, som forfatterne ikke tager højde for, er, at operatøren ikke har mulighed for at påvirke beslutningsprocessen. Dette kan være problematisk, eftersom det stort set er umuligt at opsamle al den menneskelige intuition i en model.

2.1.4 A Graph Theoretical Approach to the shunting problem

Stefano et al. [SK03] beskriver en grafteoretisk tilgang til shunting problemet. Deres fokus ligger i selve parkeringen af togenhederne, så de problemfrit kan sendes i drift den efterfølgende morgen. Herunder undersøger forfatterne de betingelser, som gør shunting problemet komplekst. De ser bort

[5] En krydsning forekommer på et depotspor, hvis et togsæt *i* hindrer ankomsten eller afgangen for togsæt *j*.
[6] restriktioner hvor der er en sammenhæng, kan samles i en clique ulighed.

fra størrelsen af toget og længden af sporet og fokuserer i stedet på betingelserne for typen af depot og den frekvens, som togene ankommer og afgår med ifølge tidsplanen. Artiklen har en teoretisk tilgang til problemet, hvor der skelnes mellem to typer af depoter hhv. en Marshalling yard, hvor sporet kan tilgå fra begge sider og en shunting yard, hvor sporet kun kan tilgå fra en side.

Som udgangspunkt antager forfatterne, at den første afgang forekommer efter den sidste ankomst, da dette demonstrerer deponeringen af togsæt over natten. Tilgangen er en minimering af antal depotspor, der skal benyttes. Der opstilles fire scenarier med differentierede tilgange til at tilgå depotet. Alle 4 scenarier udspringer af frie depotspor, hvor problemet beskrives som enten et SISO, DISO, SIDO eller DIDO problem[7]. Ved hjælp af konstruerede grafer forsøger forfatterne at finde en acceptabel løsningsmodel til de fire problemer.

Vi har nu vendt blikket væk fra den globale løsningsalgoritme, eftersom vi ikke fandt det tilstrækkeligt attraktivt i forhold til DSB S-tog. Denne artikel afspejler atter en to-trins løsning, men hvor fokus er rettet mod at skabe en solid interaktion mellem hægtningen og parkeringen. Herved forsøges der at skabe en fornuftig balance mellem en sammenkædning af indkomne og udgående togsæt. Ved at sikre dette, vil succesraten for en løsning stige. Dette er en tilgang, som vi vil tage udgangspunkt i. Ulempen bliver imidlertid, at vi ikke længere minimerer antallet af togopsplit, hvilket er et fokusområde i de tidligere nævnte artikler. Det er vigtigt ikke at forkaste problematikken omkring opsplit af indkomne togsæt, men vi vil senere i afsnit 5.1 se på, hvorledes denne restriktion på anden vis kan inddrages. Vi så eksempelvis tidligere i [LFKW03], hvordan operatøren blev inddraget i processen, og samme tilgang kan anvendes her. Dog er det en afvejning af, hvad der har den største værdi; minimering af split eller en øget chance for en brugbar løsning. Netop denne afvejning er svær at måle på og generalisere i en model, hvorfor den menneskelige operatør kan spille en vigtig rolle her.

Forfatterne tilgår problemet ud fra et ønske om teoretisk at efterprøve forskellige tilgange til et komplekst problemfelt. Eftersom vores opgave tager udgangspunkt i KH, ønsker vi kun at tage udgangspunkt i de aktuelle depotspor som beskrives i afsnit 4.2.2. Herved må vores fokus være at påvirke andre parametre i løsningen af parkeringsproblemet. Dette aspekt vender vi tilbage til i afsnit 5.1.

[7] SISO betyder, at sporet kan tilgås fra den ene side og afgå fra den anden. DISO betyder, at sporet kan tilgås fra begge sider, men kun afgå fra den ene. SIDO betyder, at sporet kan tilgås fra en side, men afgå fra begge sider. DIDO betyder, at sporet kan tilgås fra begge sider, samt afgå fra begge sider.

Afslutningsvis ønsker vi at inddrage yderligere to artikler, som hver beskriver, hvorledes den menneskelige aktør kan inddrages.

2.1.5 Decision support in the train dispatching process

Kvist et al. [KHSB02] beskriver, hvorledes et BSS kan supporte den daglige materieldisponering. I et større samarbejde forsøger de at klarlægge, hvorledes man kan udvikle et BSS, som kan hjælpe operatøren i beslutningen omkring materieldisponering i realtid. De lægger stor vægt på vigtigheden i at forstå, hvorledes operatøren arbejder, og hermed hvordan de håndterer konkrete fejl i deres re-planlægning. På denne måde ønsker de at udvikle et simpelt og let håndterbart hjælperedskab.

Det fundamentale problem i kontrollen af togdriften er, at der ofte forekommer forstyrrelser i driften. Der skelnes mellem primære og sekundære forsinkelser, hvor en primær forsinkelse er en direkte effekt af den faktiske forstyrrelse, mens de sekundære forsinkelser er et resultat af de toge, som påvirkes af den primære. I sådanne situationer er det operatørens opgave, at minimere påvirkningen af de sekundære forsinkelser ved b.la. at ændre på den oprindelige tidsplan. Denne proces bliver udført manuelt, og det er her BSS vil kunne understøtte operatøren til at tage korrekte og givetvis optimale beslutninger, når han skal foretage ændringer i tidsplanen.

For at blive klogere på selve beslutningsprocessen går forfatterne ind og kigger på samfundet omkring et BSS, hvor de erfarer, at fokus synes at være rettet mod ledende medarbejdere og støtte til deres beslutninger. I artiklen introducerer de også et andet område, som omhandler dynamiske beslutningsprocesser og handler om genopretningsplaner i realtid ved forstyrrelser.

Artiklen har fokus på at lave et BSS for genopretningsplaner i realtid, hvorimod vi ønsker at kigge på det stationære problem ved dag-til-dag planlægning. Til trods herfor har artiklen en række vigtige elementer, som vi har ladet os inspirere af. De påpeger eksempelvis vigtigheden i at inddrage den viden og erfaring, som de menneskelige aktører, der skal kunne varetage et BSS, ligger inde med. Derfor vil vi indsamle viden om den planlægningsproces, der ligger bag dag-til-dag planlægningen for at kunne imødekomme de komplikationer, der kan opstå i den samlede proces. Dette beskrives i afsnit 3.6.

Artiklen har endnu ikke forsøgt at implementere deres systemer, men henviser til fremtidigt arbejde. Herved kan det være svært at skildre, hvorvidt de fanger alle aspekter i et BSS.

2.1.6 Cognition, tasks and planning: supporting the planning of shunting operations at the Netherlands Railways

Wezel et al. [WJ08] beskriver, hvorledes det er muligt at supporte planlægningen af shunting processen i Hollands togindustri. Igennem artiklen argumenterer de for, at konstruerede algoritmer skal afspejle operatørens delopgaver fremfor det samlede planlægningsproblem. De tager udgangspunkt i den menneskelige operatør som et kognitivt system, hvorefter de kombinerer tre områder: kognitive aspekter ved den menneskelige operatørs opgaver; computersupport og algoritmer for at udvikle et planlægningsstøttesystem.

På baggrund af 15 års forskning mener forfatterne, at planlægning er så komplekst og inkorporeret i organisatorisk og kognitive problemstillinger, at der på trods af den kraft modeller og algoritmer kan bidrage med, ikke findes en matematisk tilgang til planlægning, der er tilstrækkelig på egen hånd.

Forfatterne fokuserer på den planlægning, der forekommer, når togsæt ankommer til en endestation ved slutningen af dagen og skal parkeres på et depotspor. Her skal operatøren planlægge, hvilke spor der skal benyttes; bevægelserne på sporet; hvilke køremænd der skal udfører rokeringerne på sporene; togsættenes hægtning den kommende dag samt på hvilke tidspunkter, togsættene skal rengøres.

Men hvor opstår problematikken ved automatisk genereret planlægning? Forfatterne påpeger, at tidligere undersøgelser viser, at automatisk genererede tidsplaner ofte skaber meget lidt plads til menneskelig kontrol. Samtidig viser forskning indenfor menneskelige faktorer i operationel styring, at analytiske modeller ikke kan håndtere tilstrækkelig usikkerhed og ustabilitet i den virkelige verden. Forfatterne ridser kort tre hovedfilosofier op omkring generering af en plan.

1. Computersupport, som ikke er betinget af den menneskelige operatør. Der tages ikke højde for, hvorledes operatøren løser opgaven, men derimod sættes begrænsningerne på baggrund af domæneenheder (ex. kapacitet, vagtskifte, historisk data osv.).
2. Her analyseres viden af den menneskelige operatør, som efterfølgende implementeres i et

computerprogram. Ulempen er, at løsningen afspejler den typisk menneskelige brandslukningsmetode.

3. Her etableres en koalition mellem computer og operatør, hvor man sætter operatørens opgave i fokus. Dette fremstår som den almindelige BSS opfattelse, hvor operatør og computer udfører de opgaver, de hver især er bedst til.

Forfatterne lægger stor vægt på, at det ikke er muligt at designe et planlægningsstøttesystem uden at analysere operatørens løsningsmetoder, da det vil synliggøre de delopgaver, der ligger i det samlede problem.

Gennem resten af artiklen tager forfatterne fat i en beskrivelse af deres case samt en analyse af selve opgaven for operatøren. Til sidst forsøger de at implementere en prototype, som består af flere algoritmer, der understøtter den hierarkiske struktur på de forskellige niveauer. På denne måde er det muligt, at afvikle algoritmen individuelt, mens det samtidig åbner op for en eksekvering af delopgaverne under forskellige strategier.

De tanker og teoretiske tilgange, som [WJ08] synliggør, ligner i høj grad [KHSB02]. Artiklen ligger stor vægt på, at problemstillinger af en vis kompleksitet ikke kan løses udelukkende af et maskinelt system, som kan substituere den menneskelige planlægger. Iagttagelserne om at udnytte egenskaberne for hhv. planlægger og software, samt at kunne påvirke algoritmerne undervejs i processerne ved planlægning og genplanlægning, vil vi ligeledes benytte os af.

2.2 Beslutningsstøttesystem

Vi har nu set på flere forskellige videnskabelige tilgange til problemstillingen for shunting problemet. Vi har gennem artiklerne bl.a. set, at [FLKH02] og [SLK05] slet ikke betragter eller inddrager den kognitive operatør i deres modelgenerering, hvilket vi ser belyst i de resterende artikler, som en nødvendighed i den samlede løsning af TUSP. For at vores prototype skal kunne anvendes, som en integreret del i planlægningsfasen hos DSB S-tog, finder vi det derfor yderst vigtigt at inddrage dette aspekt. Derfor vil vi uddybe forståelsen omkring det BSS, som vi vil opstille.

Grundlæggende er et BSS et computerbaseret, interaktivt system, som indsamler den nødvendige information fra forskellige kilder og dermed skaber et overblik til brugeren, som nu kan analysere og

evaluere den løsning/beslutning, der skal tages [KHSB02]. I de fleste tilfælde vil målet med et BSS ikke være at opnå en komplet løsning, men langt mere at skabe en støtte til beslutningstageren i vedkommendes beslutningsproces.

Der er foretaget mange studier af automatiseret planlægning, men når det kommer til udførelsen i praksis, går det ofte galt. Ifølge Wezel el al. [WJ08], som vi omtalte i afsnit 2.1.6, vil der aldrig være en fuldstændig løsning til et BSS. Det betyder, at når generiske modeller for opgaveløsning og computersupport dannes, sker det med det resultat, at der mistes en del information, som den menneskelige operatør besider.

Med primær inspiration fra [WJ08] vil vi koncentrere os om planlægningsprocessen, hvor der hægtes togsæt til togløb hos DSB S-tog. Vi vil i den sammenhæng fokusere på den menneskelige operatør og hans opgaver og forstå hvilke problemer, han står overfor. En dybere analytisk gennemgang af planlægningsprocessen ses i afsnit 3.6 og skal bruges til at nedbryde det samlede planlægningsproblem. Hermed bliver det muligt at danne operationelle algoritmer, der kan supporte de enkelte delopgaver.

Der findes mange automatiske løsninger for diverse problemer, men når det kommer til TUSP planlægning, er det ikke nær så udbredt. Dette skyldes, at der via de automatisk genererede planer ikke skabes plads til menneskelig kontrol, hvorfor de analytiske modeller kan have svært ved at håndtere tilstrækkelig usikkerhed og ustabilitet i den virkelige verden.

[WJ08] antyder, at den optimale proces til løsningen af TUSP forekommer ved en koalition mellem computer og bruger. Her gælder det om at udnytte hver delkomponent til det, de er bedst til. På den måde kan man udnytte det samlede automatiserede overblik, som computersupporten kan fremstille, samt den menneskelige viden, intuition og tidligere erfaringer, som operatøren ligger inde med. Vores matematiske opgave bliver således at opstille to modeller, hvormed vi bevarer det samlede overblik og sikrer muligheden for manuelle justeringer og input i løsningsprocessen. [WJ08] henviser til følgende metoder, hvormed operatøren kan inddrages i algoritmerne:

- Operatøren kan vælge mellem et antal alternative løsningsmuligheder
- Operatøren kan ændre parameterværdierne i algoritmen

Det er derfor vigtigt at forstå de forskellige delprocesser, som forekommer, når operatøren skal foretage selve planlægningen af TUSP for således at bibeholde den viden, som allerede eksisterer i DSB S-tog. Samtidigt vil vi skabe muligheden for, at operatøren kan blive præsenteret for et antal alternative løsninger. Dette skal gøres ved, at planlæggeren manuelt bidrager med viden og erfaringer til programmet for eksempelvis at afværge flaskehalse for omdirigeringer til vedligeholdelse. Samtidig, hvis modellen har svært ved at finde en brugbar løsning, skal operatøren have mulighed for at relaxere nogle af de skarpe restriktioner, således at der findes en løsning, som erfaringsmæssigt kan lade sig gøre, men som måske ikke er globalt optimale.

Til sidst i dette afsnit ønsker vi at redegøre for det software, vores beslutningsstøttesystem er bygget i.

2.3 Valg af software

Vi har valgt at benytte SAS Institute (*SAS*) som softwarepakke til løsning af vores hægtnings- og parkeringsproblem og vil i det kommende afsnit beskrive rammerne omkring den algoritme, der løses i SAS.

2.3.1 Solveren i SAS

SAS er et amerikansk softwareprogram, som blev udviklet tilbage i 1976. Dengang var SAS udelukkende et statistikprogram men er i dag bredt sig ud over en række emner bl.a. *business intelligence, customer relationship management, data mining, risk management og analytics.*

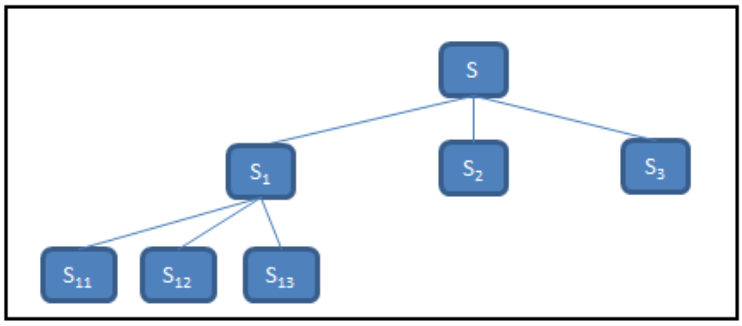
SAS tilbyder brugeren et bredt udvalg af avancerede analyseværktøjer, som man vha. forskellige statements kan påvirke. Disse programmer kaldes procedurer, og vi vil benytte proceduren Proc Optmodel. Information er hentet via [SAS11]

Proc Optmodel kan benyttes til løsning af en række forskellige matematiske programmeringsproblemer, heriblandt *Linear Programming (LP), Quadratic Programming (QP), General Nonlinear Programming (NLP) og Mixed Integer Linear Programming (MILP).* Da vores opgave kan beskrives som et lineært og binært problem, har vi valgt at benytte solveren til MILP.

2.3.2 Branch and Bound strategi

MILP solveren er baseret på en lineær Branch and Bound (*BnB*) algoritme, hvorved det reelle problem bliver opdelt i mindre delproblemer, der både er lettere og hurtigere at løse.

Hvis udfaldsrummet defineres S og deles op således at $S = S_1 \cup S_2 \cup \ldots \cup S_k$, findes den optimale løsning til hvert delproblem, og vi skal således blot udvælge den mindst mulige kriterieværdi, som derved bliver en global optimal løsning.



Figur 2 1 : Opdeling af S og de dertil hørende delmængder i Branch and Bound træet.

Opdelingen kaldes for et BnB-træ, og hver forgrening går ud til en såkaldt node. Opdelingen giver en stor mængde af delproblemer, som hver især principielt skal løses. Men heri opstår synergieffekten i træet som et resultat af, at vi ikke løser delproblemerne simultant, men sekventielt. Hermed opstår en viden omkring en optimal løsning i en beregnet node, som kan benyttes til at udelukke andre forgreninger, og på den måde kan man undgå samtlige beregninger i træet. Dette gøres ved hele tiden at forbedre den øvre og nedre grænse givet den fundne information ved løsninger til delproblemerne. Der vil nu være tre scenarier for, hvornår vi kan udelukke dele af mulighedsområdet:

- Ingen mulige løsninger til delproblemet, $S_i = \{\emptyset\}$
- En optimal mulig og heltallig løsning til S_i
- Der findes et z, der er bedre end det bedst mulige z_i i den nuværende forgrening.[8]

[8] på nuværende tidspunk Keiding

Det er hverken tilfældigt eller ligegyldigt, hvordan udvælgelsen i rækkefølgen af hvilke noder, der beregnes først, foregår. Derfor opstår der flere forskellige strategier, der hver har deres fordele (og ulemper). Herved har SAS en række tilgange, som kan påvirkes. Vi har set nærmere på følgende:

- *Automatic:* Tillader solveren selv at vælge den bedste strategi baseret ud fra problemets karakteristika og søgeproces. Denne er default i SAS. Ulempen er, at SAS er nødsaget til at genberegne den inverse del af basismatrixen i store dele af træet. Desværre er sådanne beregninger forholdsvis "dyre" i programmeringsforstand.
- *Bestbound:* Udvælger den node med mindste (eller største i tilfælde af et maksimeringsproblem) relaxerede objektværdi. Denne strategi reducerer antallet af noder, som skal gennemløbes samtidigt med hurtigt at forbedre den nedre grænseværdi. Problematikken opstår, hvis der ikke findes en god øvre grænseværdi, da det medfører at antallet af aktive noder kan blive stort.
- *Bestestimate*: Udvælger den node med mindste (eller største i tilfælde af et maksimeringsproblem) objektværdi af den estimerede heltalsløsning. Udover at forbedre den nedre grænseværdi, forsøger denne strategi ligeledes at give en god brugbar løsning.
- *Depth:* Udvælger den node, som er længst nede i søgetræet. Denne søgning er effektiv til at finde brugbare løsninger, eftersom mulige løsninger ofte længst nede i træet. Antallet af aktive noder er generelt lav, men er på bekostning af, at vi søger mere lokalt, hvilket sænker chancen for en global optimal løsning.

Desuden har SAS yderligere en række hjælpestatements, som kan anvendes til yderligere at påvirke S. Her har vi testet, om følgende statements har haft en betydning for løsningsalgoritmen:

- *Presolving: Har til formål at tætne eventuelle overflødige bibetingelser*
- *Cutting planes: Medvirker til at beskære mulighedsområdet ned til en relaxeret udgave, hvor der kun er heltalsløsninger tilbage.*
- *Primalin: Kan bruges til at "indtaste" startværdier for solveren i SAS, en såkaldt "Warm Start".*

- *Primal Heuristik: medvirker til hurtigt at finde heltalsløsninger tidligt i BnB-træet, og herved hurtigt at kunne forbedre den øvre grænse ved minimeringsproblemer. Dette medvirker til, at SAS relativt hurtigt kan reducere antal beregninger af noder.*

Både udvælgelsen af nodevalg og tilføjede statements i SAS vil vi i afsnit 5.1 teste på for at se, om vores opstillede modeller kan effektiviseres alene ud fra den algoritmiske tilgang til løsningen.

3. Køreplanlægning i DSB S-tog

Dette kapitel omhandler de bagvedliggende tanker for køreplanlægningen for DSB S-tog samt en række af faktuelle pointer omkring DSB S-tog som virksomhed. Disse skal bidrage til en hjælpende forståelse for de overvejelser, som har skabt rammerne for vores beslutningsstøttesystem.

3.1 Introduktion til DSB

DSB S-tog A/S er et datterselskab af koncernen DSB, som er ansvarlig for driften af den kollektive togtrafik (og i specialtilfælde togbusser) i Storkøbenhavn og omegnskommuner. DSB, Danske Statsbaner, er en selvstændig offentlig virksomhed, 100% ejet af Transportministeriet.

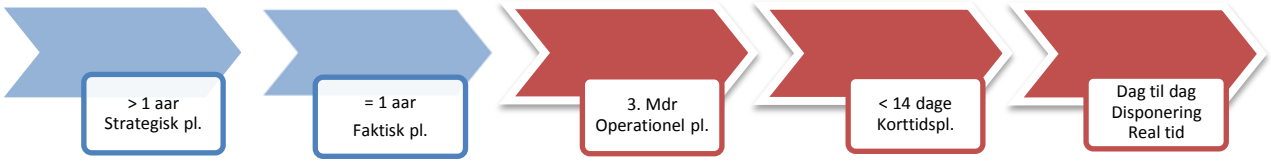
S-togsnettet består af 170 kilometers dobbeltspor og 85 stationer. Netværket er konstant optaget af omkring 80 toge gennem dagen, hvor der dagligt befinder sig 1.100 afgange fra endestationerne. Den samlede togdrift er opdelt i to selvstændige virksomheder, en operatørvirksomhed, DSB S-tog, og en infrastrukturvirksomhed, Banedanmark. DSB S-tog har ansvaret for togtrafikken, herunder at planlægge og implementere tidsplaner for togene samt opretholde en kvalitetskontrol og vedligeholdelse ud fra fastlagte rammer. Det personale, der benyttes i S-togene, bliver ligeledes planlagt og fastlagt af DSB S-tog. Ansvaret for vedligeholdelse af spor, signaler og sikkerhedssystemer etc. ligger derimod hos Banedanmark.

3.2 Formål med køreplanlægning

Det overordnede formål for DSB S-tog er at opfylde efterspørgslen på togsæder og tilgængelighed tilstrækkeligt, alt imens det forsøges at minimere de operationelle omkostninger. Dette gøres ved at tildele et optimalt antal togsæt ud fra et forventet forecast, der kan dække efterspørgslen. Togtrafikken er stærkt påvirket af en lang række restriktioner såsom statsdannede trafikkrav om tilgængelighed for offentligheden, mål for at overholde visse service- og sikkerhedskrav blandt andet. Her forsøges, der at give et overblik over hvilke problematikker og udfordringer, der gør sig gældende for DSB S-tog.

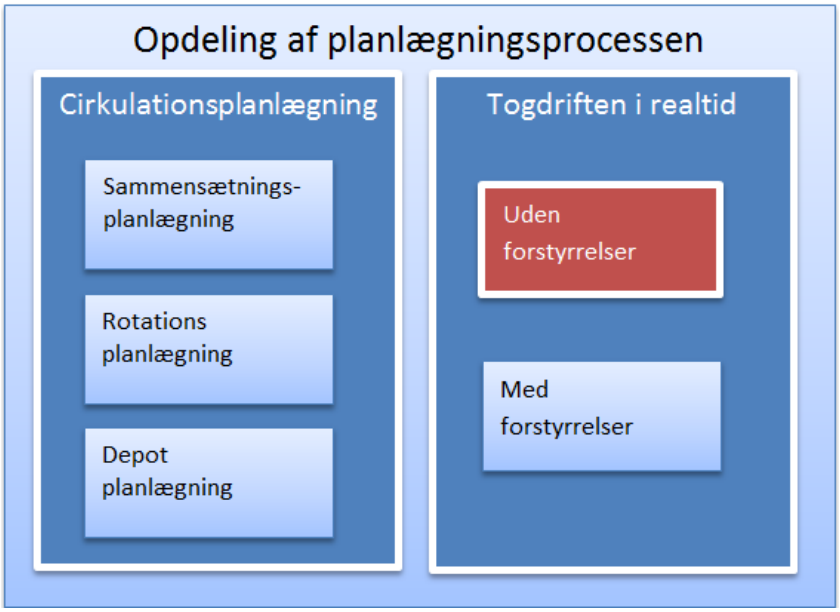
3.3 Planlægningsprocessen

Planlægningen er en omfattende on-going proces, som har start mere end et år før den faktiske udrulning, og gennemløber nedenstående proces:



Figur 3.1 : viser den samlede planlægningsproces, der foregår fra begyndelse til udrulning.

Det fremgår af figur 3.1, at den strategiske planlægning, hvor de overordnede målsætninger og problematikker kortlægges, starter mere end et år inden udrulning. Herefter påbegyndes den faktiske planlægning med et år tilbage. Begge disse delprocesser er med til at sætte restriktioner for køreplanen, men er ikke et område, som vi kan påvirke. Derfor vil vores fokus indskærpes til de sidste tre processer, som er markeret på figur 3.1 og som yderligere er opsplittet herunder.



Figur 3.2 : Oversigt over processer der indgår i planlægningsprocessens sidste 3 elementer fra Figur 3.1.

På figur 3.2 er der skitseret fem opsplitninger af den tilbageværende planlægningsfase, hvor elementet under planlægning af togdrift uden afbrydelser er særligt markeret. Dette skyldes, at vores beslutningsstøttesystem (*BSS*) har til formål at supporte netop denne fase. De resterende elementer påvirker alle i højere eller lavere grad det BSS, som vi ønsker at skabe, og kræver derfor en

tilhørende forklaring. Vi klargjorde i afsnit 2.2 vigtigheden af at forstå de rammer, der skal ligge til grund for udviklingen af vores BSS, hvorfor vi finder det nødvendigt at uddybe det følgende. Enkelte elementer vendes kun sporadisk, da vi allerede har afgrænset os og derfor ikke finder det yderligere relevant.

3.3.1 Cirkulationsplanlægningen

Ifølge den nuværende planlægningsproces skal denne del påbegyndes mindst 3 måneder før og afsluttes senest 3 uger inden udrulning. I denne proces er det ikke nødvendigt at tildele fysiske togsæt til faktiske togløb, her sættes i stedet virtuelle enheder til at varetage kørslerne. Dette skyldes, at det på nuværende tidspunkt er umuligt at sige, hvor i systemet de fysiske togsæt befinder sig 3 måneder fremme. Der kan være foretaget ad hoc-ændringer i køreplanen, sket uforudsete vedligeholdelseseftersyn blandt andet. Derfor er cirkulationsplanen blot en virtuel skitse over tognumre, der skal varetage faktiske togløb.

På nuværende tidspunkt udføres cirkulationsplanlægningen ud fra 3 mindre delprocesser som i det følgende vil blive gennemgået.

Sammensætningsplanlægning

Denne delproces beslutter hvor mange virtuelle enheder, der skal tildeles en togforbindelse for at kunne tilfredsstille efterspørgslen. De vigtigste krav i denne proces er infrastrukturen, køreplanen, tidsplanen og passagerefterspørgslen. Outputtet for denne proces er en oversigtsplan, der fortæller, hvor mange virtuelle togenheder, der skal bruges for en samlet togforbindelse, hvor det sikres, at det samlede togsæts længde ikke overstiger perronlængden, samt at hvert togsæt kun optræder en gang i køreplanen til at dække efterspørgslen. Denne del afspejles i høj grad i vores modellering under hægtningsprocessen, hvor vi tildeler fysiske enheder til de virtuelle kørsler.

Rotationsplanlægning

Her foregår beslutningsprocessen omkring hvilke virtuelle togsæt, der skal varetage de ønskede togløb. Ofte vil togsæt, som ankommer til en endestation blot vende retning og fortsætte løbet.

Depotplanlægning

Den sidste proces indebærer beslutningsprocessen for parkering og deponering af virtuelle togsæt. Dette forekommer om natten samt i dagtimer, såfremt et togsæt ikke benyttes. Dertil kommer også,

hvornår togsættet igen skal tages i brug. De vigtigste krav i denne proces er infrastrukturen ved depotsporerne samt bemandingen af mandskab ved depoterne.

Trin 2 i vores model ser nærmere på parkeringerne, der forekommer om natten, og foretager både parkeringer af de virtuelle tognummer samt de fysiske enheder.

3.3.2 Aspekter af togdriften i realtid

Togdriften i realtid kan ses som den fysiske udrulning af den virtuelle plan fra rotationsplanlægningen. Således kræves det, at de fysiske togsæt varetager den efterspørgsel, som er pålagt tidligere. Tidshorisonten for den fysiske proces er kortvarig og indgår oftest i dag-til-dag planlægningen. Dette skyldes den konstante usikkerhed i driften, hvor uforudsete hændelser kan medføre forsinkelser og ændringer i antallet af togsæt til rådighed.

Derved er togdriften i realtid opdelt i 2 elementer, som hver påvirker driften på forskellige tidspunkter under forskellige scenarier.

Uden forstyrrelser

Planlægningsfasen, som udspiller sig i denne situation, er målsætningen for vores BSS. Den generelle tankegang (og formål) er, at de krav, der allerede håndteres i cirkulationsplanlægningen, ikke behøves at blive behandlet under togdriften i realtid, såfremt der ikke optræder forstyrrelser i driften. Det betyder, at vores tilgang til løsningen i denne fase er at hægte de allerede fastlagte tognumre til de fysiske togsæt.

Med forstyrrelser

I virkeligheden sker der ofte både mindre og større forstyrrelser, som har indflydelse på den planlagte drift. Nogle forstyrrelser er tydelige og medfører deciderede aflysninger eller omdirigeringer og andre er mindre synlige. Vi kender det fra efterårsstorme, som nedlægger driften i kortere og længere perioder, signalfejl hvor vi må afvente videre drift eller mange andre scenarier, hvor den oprindelige køreplan ikke kan overholdes. En *robust* cirkulationsplan er en plan, som kan håndtere udefrakommende driftsforstyrrelser ved mindre justeringer af den originale virtuelle plan, hvor det i takt med ændringerne gælder om at minimere de negative omkostninger.

Når der forekommer forstyrrelser i køreplanen, er det DSB S-tog og Banedanmarks opgave at genoprette den driftsmæssige køreplan i fællesskab. Dette skal foregå så hurtigt og omkostningsfrit

som muligt. Og netop hastigheden af genoprettelsen er ofte det primære mål, eftersom forstyrrelser kan sprede sig som ringe i vandet til resten af netværket.

Vi vil ikke håndtere situationer med forstyrrelser og genopretningsplaner yderligere, men det er et vigtigt element at have for øje, når vi skal lave et støttesystem til den normale drift. Dette skyldes, at vi skal sikre at lave en så robust løsning som muligt, således at ringenes spredning kan begrænses mest muligt.

3.4 Omkostninger forbundet med køreplanlægningen

For at kunne skabe en køreplan for den samlede drift er det nødvendigt, at de enkelte togsæt er placeret korrekt i netværket, således at efterspørgslen kan dækkes. For at opnå dette vil der være knyttet forskellige omkostninger i forhold til typen af positionering, der sikrer et samlet flow i netværket. Baggrunden for omkostningerne ligger i høj grad til grund for, hvorfor vi ønsker at kunne foretage omdirigeringer undervejs i den normale drift for ikke at få pålagt for mange ekstra omkostninger.

Formålet for DSB S-tog er som udgangspunkt at sikre, at efterspørgslen dækkes men med fokus på at minimere omkostningerne. Til tider vil det kunne svare sig at påtage sig meromkostninger for en hurtig positionering for at nedsætte belastningen af netværket, men ofte er der et ønske om at foretage disse positioneringer i takt med den planlagte drift. Der er således to scenarier for positionering, som vi kan opstille:

16. En flytning af en enhed, som undervejs yder en togservice til passagerer, hvilket kaldes for en *indtægtsgivende positionering.*

17. En flytning af en enhed, hvortil det ikke er muligt for passagerer at benytte togsættet. Dette kaldes en *ikke indtægtsgivende positionering.* Her kan der være tale om situationer, hvor der ikke er tilstrækkelige togsæt på KH til at varetage driften.

Dette kan bl.a. komme til udtryk i køreplanen, hvor en indtægtsgivende positionering består af en overdækning på en efterspørgsel med flere togenheder eller en omdirigering til servicelinjen pga. vedligeholdelsestjek.

3.5 Praktiske foranstaltninger for køreplanen

Dette afsnit beskriver hvilke rammer, der er forbundet med at lave en køreplan. De krav (restriktioner), som opstilles, kan inddeles i to overordnede kategorier, som enten er afhængig eller uafhængig af de individuelle fysiske togsæt.

3.5.1 Krav uafhængigt af de enkelte fysiske togsæt

Størstedelen af disse restriktioner håndteres i cirkulationsplanlægningen, hvorfor de indirekte påvirker vores model.

Infrastruktur

Et af de vigtigste krav, der ligger i køreplanlægningen, er infrastrukturen. Infrastrukturen består af alt fra spor, knudepunkter, platforme, depoter, værksteder osv.. Langt de største restriktioner er der taget højde for i cirkulationsplanlægningen, men enkelte elementer sætter også begrænsninger i vores modeller. Her kan bl.a. nævnes det begrænsende antal af toglinjer, som fører mod værkstedet eller begrænsningerne af depotstrukturen, som vi vender tilbage til i afsnit 4.2.2.

Tilgængelighed

DSB S-tog er af transportministeriet pålagt nogle restriktioner og formelle krav omkring tilgængeligheden af offentlig togtransport. Derfor er selve planlægningen af togdriften afhængig af den fastlagte køreplan, som viser hvor og hvornår, togene vil gøre holdt på de enkelte strækninger.

Den nuværende køreplan iværksættes således, at den er opdelt i hverdage og weekenddage. I hverdagene kører den linje H og Bx hvert 20 min. F linjen kører hvert 5. min. Alle andre linjer kører hvert 10. min. På figur 3.3 herunder vises den samlede S-togsdrift. Hvis der eksempelvis er indtegnet et linje-bogstav inde i selve linjen, som det ses for linje A, hvor der findes et A mellem Hundige og Greve, betyder dette, at det varierer, hvornår toget kører helt til endestationen Solrød Strand. Det fremgår ligeledes at der findes 6 forgreninger samt en ringbane, der går fra Hellerup i nord til Ny Ellebjerg i syd.

Figur 3.3 : Overblik over køreplan for alle toglinjer i København og omegn. Hver linje er knyttet op med et bogstav og en farve. Denne køreplan er den primære køreplan, der hovedsagligt benyttes i dagtimerne i hverdagene.

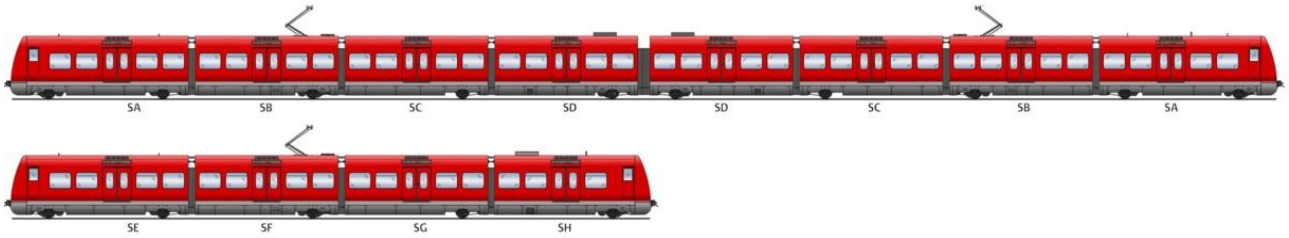
Det fremgår af figur 3.3 at linjerne A, B og Bx kan betragtes som servicelinjer, da de vil føre togsættet ud til hhv. Hundige (*UND*) og Høje Taastrup (*HTÅ*), hvor netværkets værksteder er placeret.

Togsæt

DSB S-tog har 2 typer af togsæt, som betegnes *Litra SE* og *Litra SA*, eller alternativt LHF og LHB. Andre informationer om de 2 typer af togsæt ses i tabel 3.1.

Type	Teknisk beskrivelse	# af pladser teoretisk	# af pladser praksis	# vogne	Længde af togsæt [m]	# af togsæt DSB Stog total
½ (LHF)	Litra SE	150	125	4	42,58	31
1 (LHB)	Litra SA	366	300	8	83,78	104

Tabel 3.1 : Tabeloversigt der viser specifikationer for de 2 typer af togsæt DSB S-tog figurerer med.



Figur 3.4 : øverst LHB tog og nederst LHF.

Udseende for de to typer af togsæt fremgår af figur 3.4 og tilsammen kan disse sammensættes via fem forskellige kombinationer betinget af to forskellige restriktioner, se tabel 3.2. Ved en kombination af mere end en enhed kaldes flere togsæt for en togkomposition. I det følgende beskrives restriktionerne, som er gældende for en togkomposition.

1) Længden af perronen.

For et indtægtsgivende tog er der et krav om, at togkompositionens længde ikke må overstige perronlængden, eftersom alle døre skal kunne åbnes og benyttes til på- og afstigning. Dertil er stationerne inddelt i to grupperinger, hvor stationerne på F-linjen kun har kapacitet til kombinationer af LHF, mens det resterende netværk kan håndtere op til to LHB enheder.

2) Antal togsæt i en togkomposition

Så længe vi befinder os i planlægningsfasen, må antallet af togsæt i en togkomposition ikke overstige to enheder af gangen. Dette er en af de mere hyppige metoder til at korrigere køreplanen, hvis der er opstået forstyrrelser, og den planlagte trafik skal genoprettes. En konsekvens af en togkomposition af mere end to enheder kan være, at nogle af togsættene kører passivt med, hvilket betyder en langsommere acceleration. Andre fejl kan betyde manuelt arbejde for togføreren, hvilket er tidskrævende. Det er derfor hverken robust eller optimalt at køre med ekstra togsæt, men bliver som sagt praktiseret.

I tabel 3.2 ses de 5 kombinationsmuligheder af togsæt

Kombinations-muligheder	Togenheder	# pladser teoretisk	Kørsel
½	½	150	F linje
$^{2}/_{2}$	½+½	300	F linje
1	1	366	Alle undtaget F linje
1½	1+½	516	Alle undtaget F linje
2	1+1	732	Alle undtaget F linje

Tabel 3.2 : Kombinationsmuligheder for togsammensætning.

Dette leder hen til det sidste krav, som ikke er enhedsafhængigt, passagerefterspørgslen.

Passagerefterspørgsel

Som det fremgår af tabel 3.1, skelnes der mellem teoretiske og praktiske siddepladser, hvor tabellen angiver det teoretiske antal. Opdelingen sker på baggrund af et kritiske niveau, som er defineret ved, at DSB S-tog har erfaret, hvornår der sker en kraftig stigning i klager ved pladsmangel. Planlægningen sker således ud fra det praktiske antal. I vores tilgang til problemet er dette underordnet, eftersom planlægningen for at dække efterspørgslen allerede sker i cirkulationsfasen og derfor allerede er varetaget i den virtuelle plan.

3.5.2 Krav afhængt af de fysiske togsæt

Vi vender nu blikket mod de individuelle togsæt, og hvilke restriktioner hvert togsæt kan begrænses af.

Planlagte eftersyn

For at sikre kvalitet og sikkerhed har Trafikstyrelsen nedsat en række betingelser, som DSB S-tog er pålagt i form af planlagte eftersyn med faste frekvenser. Alt afhængigt af togsættes type ligger der forskellige krav for, hvornår disse eftersyn skal indtræffe. Det hyppigste sker hver 50.000 km og betegnes 50 Mm eftersyn. Dette vil vi løbende referere til som vedligeholdelseseftersynet, og kun dette er indbygget i modellen. Der optræder i alt 12 typer af eftersyn for hhv. LHF og LHB, men de resterende eftersyn udelades på baggrund af at opbygningen af restriktionerne vil være identiske. Desuden vil alle eftersyn principielt set blive indfanget ved indkaldelse hver 50 Mm.

Udfordringen omkring de regelmæssige eftersyn skabes på baggrund af den store forskel i distancen for de enkelte strækninger, der tilbagelægges. Dag til dag er der derfor en stor variation af tilbagelagte distancer. I løbet af dagen forekommer der ligeledes til- og afkoblinger, som yderligere varierer det samlede antal kørte kilometer pr. togsæt. Den gennemsnitlige strækning for LHF er 250 km dagligt og 500 km for LHB. Variationen kan også skyldes, at et togsæt nogle dage helt fritages for kørsel, hvor togsættet andre dage blot benyttes som en tilkobling i myldretiden. Helt tredje kan det være, at togsættet er den primære enhed på en togstrækning, hvilket medfører en lang driftsdistance. Yderligere forekommer der stor forskel på distancen for de enkelte toglinjer. Et togsæt der er i drift hele dagen på Køge-Hillerød (E) strækningen tilbagelægger i omegnen af 1000 km, hvorimod et togsæt, der er i konstant service på Ny Ellebjerg-Hellerup (F), blot tilbagelægger 350 km.

Ikke planlagte eftersyn

Når en enhed skal til eftersyn, omend det er planlagt eller ej, efterstræbes en løsning, hvor der laves en omdirigering tids nok til, at vi kan foretage en indtægtsgivende positionering mod HTÅ eller UND som endestation, hvor serviceværkstederne ligger. Afhængigt af omfang har DSB S-tog valgt at kategorisere fejltyperne ud fra deres varighed.

Kategori	Bekymring	Behandling	Eksempel
A	Stor	Straks	2 efterfølgende dør sæt der ikke kan åbnes
B	Medium	3 timer	Graffiti
B1	Medium	Kl. 03:00 næste dag	Defekt varmeregulering
C	Lav	72 timer	1 dørsæt, der ikke kan åbnes
C1	Lav	1 uge	Kompresser fejl til bremsning.
D	Ingen	Næste planlagte eftersyn	Sprunget lysstofrør

Tabel 3.3 : detaljeret kategorisering af uforudsete fejl samt specifikationen for, hvorvidt disse skal **behandles.**

Vi ønsker ikke at implementere de kategoriserede fejl som en automatisk proces, der sender togsættene til vedligehold. Derimod vil vi håndtere fejlene med en særlig bevågenhed, som skal skabe en mulighed for planlæggeren, hvormed han kan påvirke løsningen efter bedste intuition. Vi ønsker derfor sideløbende med den normale restriktion om vedligeholdelse, at sikre muligheden for at sende kategoriserede fejltog i bestemte togløb, så de kan blive tilset.

Dette er et forsøg på at efterligne DSB S-togs nuværende håndtering af uforudsete fejl. Her trækkes der dagligt en liste over togsæt og tilhørende fejlinformationer, og planlæggeren har så mulighed for omdirigering. Herved kan vi også lave en hierarkisk opbygning mellem planlagte eftersyn og uforudsete fejlkategorier.

I afsnit 5.1 vender vi tilbage til, hvordan sådanne hierarkiske restriktioner kunne være direkte indbygget i modellen, da det i andre situationer kan være praktisk at automatisere. Her vil vi mene, at det bliver et tab i bevågenhed, hvorfor det fravælges.

Foruden værkstedet i HTÅ og UND har DSB S-tog et mobilt mekanikerhold, som har hovedsæde i UND. Herfra kan der udsendes et hold, som kan udbedre større og mindre skader, og dermed i en ad hoc-løsning sørge for at ændre eventuelle A fejl til enten B eller C fejl, således at togsæt midlertidigt kan fortsætte sin igangværende service.

Friktionssand

En anden og betydelig restriktion er genopfyldning af sand. Friktionssand bruges som middel mod glatte togskinner i forbindelse med nedfaldne blade, frost etc. Hvis en enhed løber tør for sand, er det besluttet, at der indtræder en hastighedsbegrænsning på 60 km/t, indtil en genopfyldning er foretaget. Denne påfyldning kan ske på UND og HTÅ og skal for begge typer af togsæt foretages hver 10.000 km (10 Mm).

Der findes mange restriktioner i forbindelse med vedligeholdelsen af de fysiske enheder, men da vi ikke fokuserer på samtlige i vores model, undlades en dybere beskrivelse af disse. Dette skyldes bl.a. at varigheden, samt hvorvidt hændelsen finder sted, varierer og ofte er stokastisk uafhængig.

Denne form for vedligeholdelse indbefatter bl.a. uden- og indendørs rengøring, fjernelse af graffiti, vinterforberedelse, eksponering af reklamer, sikkerhedsovervågning, overfladebehandling osv.

3.6 Korttidsplanlægning af S-tog materiel

Vi beskrev med belæg i Wezel et al. [WJ08] og Kvist et al. [KHSB02], at et beslutningsstøttesystem ikke kan opstilles medmindre, at den verden og de mennesker, der befinder sig i den, inddrages. Vi vil i det kommende med baggrund i et møde og en rundvisning med den operative del af DSB S-togs planlægning, beskrive hvordan de nuværende processer foregår, når dag-til-dag planen udrulles.

På den korte bane foregår der et stort arbejde med tildeling, disponering og planlægning af materiel. Dette, har vi erfaret, primært udføres i S-tog kontrolcenter (DICS'en), hvor de forskellige operationelle enheder befinder sig, heriblandt:

- Materiel Styring (*MAS*): Materiel disponering og korttidsplanlægning.
- Banedanmark (*FC*): Materiel disponering
- Lokomotivinstruktører
- S-tog info (*INFO*): Information til passagerer
- S-tog personaledisponent (*bagvagt*)

DICS'en er knudepunktet for dag-til-dag planlægningen, og det er her, den daglige drift styres. Vores BSS har fokus på planlæggeren, der sidder i MAS, som udfører dag-til-dag udrulningen af de fysiske enheder.

3.6.1 Omdirigering

Når et togsæt kræver nødvendig service, så er det MAS i DISC'en, der skal sikre, at togsættet omdirigeres fra dets nuværende position til et togløb, som fører togsættet ud til enten HTÅ eller UND. Dette overblik indhentes vha. en database, som informerer MAS omkring, hvilken service de enkelte togsæt står overfor og hvornår. Når driften starter om morgenen, og togsættene er blevet sendt i de relevante togløb, er det efterfølgende værkstedernes eget ansvar at udtrække og indsætte de nødvendige togsæt. Opgaven for MAS er således at finde og kommunikere de nødvendige omdirigeringer, der sikrer, at de rigtige togsæt sendes i relevante togløb. Når MAS har indhentet det nødvendige data for at kunne afgøre, hvilke togsæt der bør sendes til service, undersøges det til hvilke togløb, de enkelte togsæt er tildelt, eller alternativt på hvilke depoter de er placeret.

3.6.2 Hægtning

I øjeblikket udfører MAS korttidsplanlægningen manuelt. Først sikres det, at antallet af tilgængelige togsæt er tilstrækkelig. Er det ikke tilfældet, skal der leveres nye togsæt fra værkstederne. Næste skridt er at hægte indkommende togsæt til togløb, og tage højde for parkeringen af de ankomne enheder. Hægtningen er begrænset af, at depotplanlægningen skal gå op. Det betyder, at det materiel, som er tildelt et udgående togløb, har fri passage uden behov for rangering af øvrige enheder.

Denne planlægning kan blive forstyrret af realtidsdisponering af materiellet, der skal foretages af FC for at modsvare uforudsete hændelser. I dette tilfælde kan MAS være tvunget til at rette sig ind, hvorfor en del af korttidsplanlægningen skal ændres tilsvarende, hvilket kan medføre genplanlægning af omdirigeringer såvel som hægtninger. Den operative planlægger skal derfor være erfaren og have en stor indsigt i problematikken, som er svær at standardisere.

Når hægtningen er fastsat på KH, sendes en rangerplan[9], der viser hægtningen mellem togsæt og togløb, til servicestationen i UND og vedligeholdelse i HTÅ. Herefter er det bemandingen på de pågældende depoter, der beslutter, hvorledes der løbende skal hægtes dagen igennem. Årsagen til

[9] Se bilag 2

dette system begrundes i, at det giver personalet på UND og HTÅ mulighed for løbende igennem dagen at udtage de togsæt, der har krav om service.

3.7 Servicelinjen

Vi har allerede nævnt, at de endestationer, der kan varetage forskellige serviceydelser, er hhv. HTÅ og UND. Vi har ligeledes fået bekræftet, at omdirigeringerne kan ske i løbet af dagen eller efter endt drift afhængigt af nødvendigheden. Til vores model har vi valgt udelukkende at have fokus på omdirigering efter endt drift, eftersom dette foretrækkes af DSB S-tog. Derved bliver løbende ad hoc-løsninger overladt til den erfarne planlægger i MAS.

Med udgangspunkt i de togløb, der har afgang fra KH, vil der i hverdagen være omkring fem togløb, der kører på B-linjen mod HTÅ, og to togløb der kører på A-linjen mod UND. Denne information er interessant, når vi skal kigge på hhv. vedligeholdelse og sandpåfyldning ved hhv. 50 Mm og 10 Mm. DSB S-tog har en målsætning om at overholde disse, men samtidig ønsker de, at materiellet skal køre så langt som muligt, inden det sendes til eftersyn. Det er derfor vigtigt, at kende til den kilometerdistance, som ligger bag de enkelte togløb, da dette reelt set er distancen, som togsættet skal kunne tilbagelægge, før det igen sættes i et nyt togløb.

Med henblik på udtagelse af togsæt til service, skal vi nu fastlægge, hvilken distance vi skal forholde os til ved kilometerbegrænsning senere hen under modellen. For at simplificere forklaringen tager vi i første omgang et kig på vedligeholdelseslinjen B. Når et togsæt skal udtages til eftersyn, vil det blive fikseret over på B-linjen, og hermed ende på HTÅ. Optimalt set behøver det enkelte togsæt kun have tilstrækkeligt tilbageværende kilometer til, at kunne tilbagelægge den distance, der er fra KH til første ankomst på HTÅ. Denne distance afhænger af, om togsættet i sit udgangspunkt er hægtet på en linje, der kører mod nord eller syd. Resultatet af dette optimum er et tab i robustheden, hvilket skal forstås som, at værkstedet kun har én chance for at udtage togsættet fra driften. I tilfælde af, at værkstedet er booket, eller der forekommer en højere prioriteret fejl på et andet togsæt, vil der opstå komplikationer. Det tilsendte togsæt kan altså ikke varetages på værkstedet, men har heller ikke mulighed for at køre videre til næste gang enheden ankommer på HTÅ, da kilometerrestriktion ikke tillader det (såfremt at togsættet først er fikseret til servicelinjen i sidste øjeblik). Dette svækker robustheden i netværket, som er et af de vigtige elementer i DSB S-togs målsætningsstrategi:

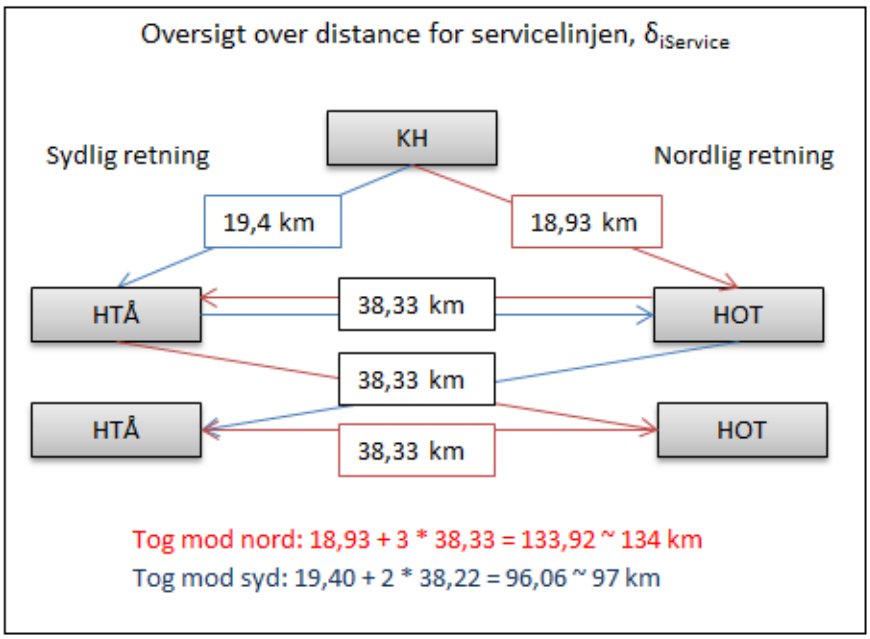
…bidrage til bæredygtig vækst og mobilitet i Danmark.

Det gør vi konkret ved at:

sikre pålidelig togdrift og sammenhængende kollektiv trafik. [10]

En pålidelig togdrift er blandt andet ensbetydende med en robust togdrift. Derfor har DSB S-tog haft et ønske om en vægtet løsning, hvor vi bibeholder en vis grad af robusthed på bekostning af et tidligere krav om fiksering af togenheden.

Efter ønske fra DSB S-tog har vi valgt at kræve, at en fiksering senest forekommer, når togsættet som minimum kan nå to ankomster på depotet, uafhængigt af, hvilken retning togsættet kører ved afgang fra KH. På nedenstående figur 3.5 ses driften mellem KH-HTÅ-HOT (Holte).



Figur 3.5 : maksimal distance til et vedligeholdelsestjek, forudsat at afgangen sker fra KH og at depotet i HTÅ er besøgt to gange. Rød rute sendes i sit udgangspunkt mod HOT og blå linje sendes direkte til HTÅ.

Vi kan se, at den længste rute, som muliggør to stop på HTÅ, er turen mod Nord på 134 km. Vi vil derfor lade en togenheder varetage B-linjen så længe, der er mere end 134 km retur inden næste eftersyn.

På samme vis, har vi kigget på distancen til UND. Både A og E linjen passerer UND, men da det skal være muligt, at udtage enheden uden større driftsforstyrrelser, ser vi kun på A-linjen. A-linjen har

[10] [DSB13], s. 13 – Den store sammenhæng

UND som endestation og vi karakteriserer den som en del af servicelinjen. I hverdagen afgår der fra depotet på KH kun A-linjer i sydlig retning, og da vi ønsker, at togsættene skal kunne ankomme to gange på UND før kilometerbegrænsningen overskrides, giver det et krav om en samlet distance på 110 km. Da denne distance domineres af kilometerdistancen for B-linjen, har vi valgt, at fastholde en samlet kilometerdistance for alle servicelinjer på 134 km.

3.8 Afrunding

Den samlede opgave, som DSB S-tog varetager, er nu beskrevet igennem dette kapitel. Vi har skabt en indsigt i det komplekse sammenspil, som ligger bag dag-dag driften, og pointeret nogle af de problematikker DSB S-tog står overfor. Med denne viden er vi nu i stand til at kaste os ud i den matematiske formulering af de enkelte delprocesser, som vi netop har gennemgået.

4. Modelgennemgang

4.1 Indledning

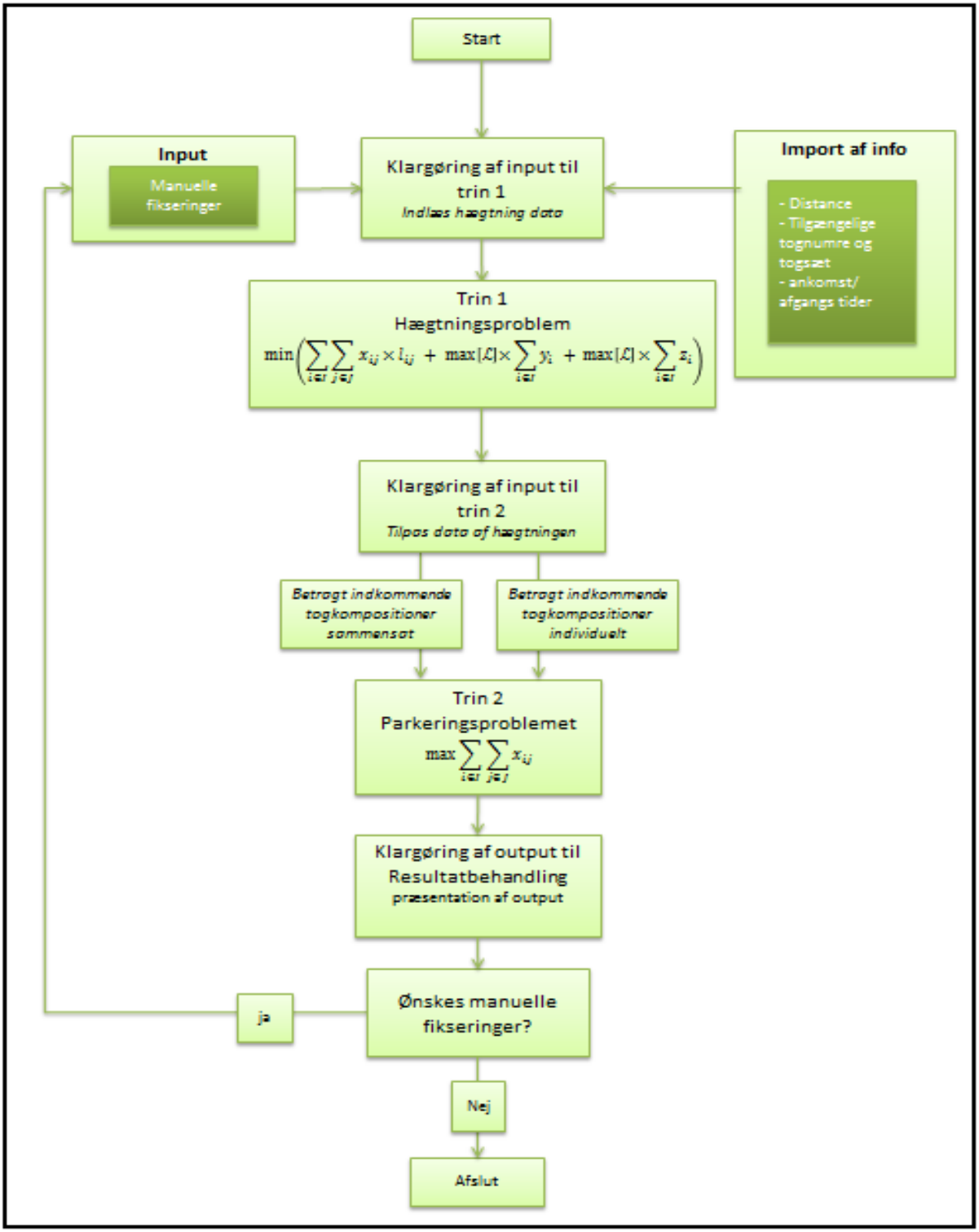
Som tidligere nævnt er vores interesse i samråd med DSB S-tog, at skabe en model, der kan varetage den fysiske hægtning mellem togsæt og de virtuelle tognumre, der er blevet dannet i cirkulationsplanlægningen. Når hægtning af togsæt har fundet sted, skal de efterfølgende parkeres, så det sikres, at næste dags udrulning af driften frit kan foretages uden omrokeringer på depotsporene.

Dette beskrives som rammerne for det klassiske shunting problem af Freling et al. [FLKH02], og vi har valgt at lade os inspirere af tilgangen til løsningen af problemstillingen, som værende en to-trins løsningsmodel af shunting problemet. Dertil har vi valgt vores egen tilgang til modellernes søgekriterier for et optimum, hvor vi benytter de restriktioner, som skaber rammerne omkring hægtning og parkering hos DSB S-tog.

Udfordringen opstår som led i et ønske om en global optimering, såfremt der findes en løsning. Dette er problematisk, eftersom vi er nødsaget til at løse begge problemstillinger simultant, hvilket betyder, at vi bevæger os i et flerdimensionelt beslutningsrum. Schrijver et al. [SLK05] viser bl.a., at resultatet af et stor udfaldsrum bliver, at sandsynligheden for at finde en mulig løsning inden for et acceptabelt tidsrum falder. Den diametrale modsætning til den globale søgealgoritme er en løsning, der findes ved en række manuelle beslutningsprocesser. Her forekommer det, at operatøren mister det brede perspektiv, hvilket resulterer i langt fra optimale beslutninger.

Dette er ligeledes et kendt problem, som også understreges i litteraturen. Wezel et al. [WB02] påpeger, at man bør inddrage både den maskinelle og den manuelle proces for at opnå en samlet optimal planlægning. Lentink et al. [LFKW03] arbejder som tidligere nævnt med en fire-trins løsning, hvor de forsøger at implementere beslutningsprocesserne i den samlede løsning, hvilket også har været en inspiration til vores fremgangsmåde.

Vi vil derfor forsøge at løse problemfeltet sekventielt i en to-trins løsning. Når løsningen er præsenteret, skal det være muligt, at foretage en manuel løsning, såfremt bruger ønsker yderligere omdirigering. Herunder vises et flowdiagram over programmets udformning:



Figur 4.1 : flowdiagram af opbygningen i SAS af valgte to-trins løsning.

Flowdiagrammet illustrerer således, hvordan programmet starter med at indhente en række informationer, som operatøren i MAS dagligt tilgår vha. deres systemer. De fleste af disse informationer er resultatet af planlægningen, som vi beskrev i afsnit 3. Her hentes eksempelvis al information ind omkring virtuelle tognumre, såsom hvilke tognumre der skal bruges, og hvornår de skal afgå. Andre informationer tilgår på baggrund af dagens drift, hvor det fastslås, hvilke togsæt der ankommer på KH, hvad tid og hvor langt materiellet har tilbagelagt siden sidste eftersyn.

Når al information er læst ind i programmet, kan trin 1 løses, hvorved vi danner en løsning af hægtning mellem togsæt til togløb. Efterfølgende klargøres output fra hægtningsprocessen, så det kan indgå i parkeringsproblemet. Her har brugeren mulighed for manuelt at vælge, hvorvidt operatøren ønsker at se indkomne togkompositioner som én parkering eller som to individuelle togsæt.

Som omtalt i afsnit 2 ønsker vi størst fokus på at finde en mulig løsning, og kun sekundært et forsøg på minimering af split. Dette giver brugeren mulighed for i første omgang at se, om det er muligt at finde parkeringer til hele løsningen, hvor vi forsøger at bibeholde indkomne togkompositioner. Herefter kan der foretages en vurdering om, hvorvidt denne løsning kan lade sig gøre. Alternativt kan der forekomme en genplanlægning.

Når vi finder en brugbar løsning til parkeringen, har vi en samlet løsning til TUSP. I tilfælde af, at løsningen ikke benytter alle tilgængelige omdirigeringer til servicelinjen pga. krav til eftersyn, vil det nu være muligt for operatøren manuelt at fiksere yderligere togsæt til disse togløb. Denne mulighed er integreret som led i at afværge eventuelle flaskehalssituationer på værkstedet, samt en løsning til at varetage de uforudsete kategoriserede fejl.

I det efterfølgende vil vi opdele afsnittet i hhv. hægtnings- og parkeringsproblemet, hvor vi vil beskrive opbygningen af modellerne.

4.2 Hægtningsproblemet

Der er omkring 30 togløb som har afgang fra KH i hverdagen. Disse løb er sat op, så de gennem deres cirkulation hovedsageligt ender på KH eller HTÅ. Fordelen herved er, at de togsæt, som har bestemte behov, enten kan dirigeres ud til HTÅ eller UND. Her kan de kan tages ud af driften i løbet af deres

rute, eller blot færdiggøre dagen og ende på et vedligeholdelsesdepot. Dette aspekt er beskrevet i afsnit 3.7.

Når et togløb starter, varierer antallet af dage fra 0 til 4 før løbet, som togsættet er tildelt, planlægningsvist slutter. Det kan derfor betyde, at når et togsæt tildeles et togløb fra KH mandag, vil det variere – givet at DSB S-tog bibeholder dette materiel i togløbet - hvornår materiellet igen ankommer til deponering på KH.

Efterspørgslen på togsæder kan kræve, at der sammensættes flere togsæt for at imødekomme antallet af passagerer på visse strækninger. I nogle tilfælde er dette kun krævet i bestemte tidsperioder, og derfor kan et tognummer forudsætte to togsæt. Når et tognummer indgår i flere togløb, er det ikke givet, at disse starter og slutter på samme station. Dette er nødvendigt at forstå, da det betyder, at kilometerdistancen for to togsæt, som sendes af sted i to forskellige togløb med samme begyndelsestognummer, kan variere. Dette kan illustreres ved nedenstående figur:

Vogn	Køredag	05.41.00	05.59.00	06.56.30	08.31.30	09.56.30	10.59.00	---	---	---
LHF	1	52117(P1)	52218(P-2)	50123(P1)	50227(P-2)	52132(P1)	52233(P-2)	50143(P1)	50247(P-2)	.....
LHB	1	52117(P2)	52218(P-1)	50123(P2)	50227(P-1)	52132(P2)	52233(P-1)	52138(P1)	52239(P-1)	.....

Tabel 4.1 : Her ses oversigt over to togløb, som indeholder samme start tognummer. Vi ser her, at den samlede række af tognumre ikke er ens for de to togløb, hvorfor de på et tidspunkt vil blive splittet op.

På figur 4.1 er der taget udgangspunkt i tognummer 52117, som afgår kl. 05.41 fra KH i hverdagen. Dette tognummer er tildelt to togsæt fra hvert sit togløb, hvor der efterspørges hhv. et LHF og et LHB. Vi ser, at togkompositionen sammenholdes henover seks tognumre før togsættene splittes op. De to togsæt forsætter nu i hvert sit togløb, og det er herved klart, at den kilometerdistance, som de to togsæt tilbagelægger før de igen er på KH, ikke er den samme. Såfremt der er to togløb, som består at samme række af tognumre fra start til slut, ses der ligeledes på to togløb, der blot har identiske tognumre igennem hele netværket.

Inden vi går i dybden med den matematiske formulering af problemet, har vi fundet det nødvendigt at redegøre for de parameterværdier, vi pålægger i vores objektfunktion.

4.2.1 Løsningsforslag med separation

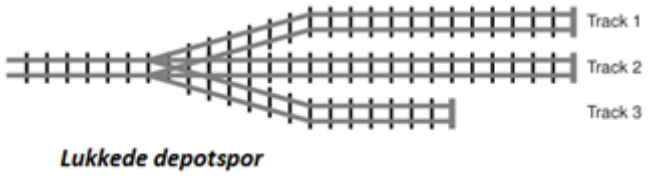
Som tidligere nævnt vil den optimale globale løsningsalgoritme bygge på et samlet problem, der både varetager parkeringen og hægtningen i en og samme model, men ofte med en uacceptabel

beregningstid. Idet vi har valgt ikke at benytte en simultan beslutningsproces, gælder det således om at sikre en solid interferens mellem de to problemer. Derfor er målet ikke bare at skabe en optimal løsning i hægtningsproblemet alene, da vi således ikke tager højde for de rammer og krav der stilles for parkeringsproblemet. Omvendt tillader en optimal parkering ikke nødvendigvis en mulig hægtning. I begge situationer kan vi ende ud i en situation, hvor vi må forkaste modellerne.

Formålet bliver således at lave to delprocesser, som i så vid udstrækning forsøger at tage hensyn til hinanden. Dette kan gøres vha. LIFO matricer, som i det efterfølgende vil blive gennemgået.

4.2.2 LIFO Matricer – fordele og ulemper

Infrastrukturen rundt i netværket består af mange forskellige typer af depotspor. I afsnit 2.1.4 beskrev vi kort, hvorledes Stefano et al. [SK03] tester på 4 forskellige sportyper til deponeringen. Vi har som primært formål valgt at beskrive og løse problemstillingen på KH, hvor depotområdet består af en enkelt type depotspor. Denne type illustreres herunder:



Figur 4.2 : viser typen af depotspor som vi tager i betragtning på KH.

Sportypen kaldes lukkede depotspor, og er de mest begrænsende sportyper for parkeringen. Dette skyldes at der kun er en vej ind og ud på sporet, hvilket også angives i navnet. Disse kaldes også LIFO spor (Last-In-First-Out), som ligeledes indikerer begrænsningen.

På KH findes der to depotområder hhv. øst og vest. Vi koncentrerer os omkring depot vest, hvor der befinder sig syv LIFO-spor til parkering. For at sikre at modellen så vidt som muligt forsøger at hægte sidst ankomne togsæt med første afgang, er vi derfor nødt til at tildele vores objektfunktion en omkostningsfaktor, som favoriserer denne hægtning. Hvis der ikke tages forbehold for dette, vil resultatet blive en stor mængde af reelle løsninger til det ordinære hægtningsproblem, hvor en fysisk parkering vil være umulig at foretage. Omvendt er det ikke et krav, at sidst ankomne togsæt skal hægtes med første afgang, da der foreligger flere depotspor på depotområdet. Det kan eksempelvis

godt lade sig gøre at hægte det næstsidste togsæt til morgendagens første togløb, såfremt at det er placeret længst ude på et af de andre depotspor.

Vi ønsker nu, at beskrive en omkostningsmultiplikator, som skal favorisere en LIFO-hægtning mellem ankomst og afgang. Multiplikatoren kan opstilles, som illustreret herunder:

	Afgang:	04:51	04:58	05:05	05:06	05:06	05:09	05:10	05:11
Ankomst	Togsæt	Togløb1	Togløb2	Togløb3	Togløb4	Togløb5	Togløb6	Togløb7	Togløb8
23:31	Togsæt 1								
23:02	Togsæt 2								
22:53	Togsæt 3								
22:48	Togsæt 4								
22:48	Togsæt 5								
22:21	Togsæt 6								
22:07	Togsæt 7								
21:41	Togsæt 8								
21:41	Togsæt 9								
21:36	Togsæt 10								

Figur 4.3 : LIFO matricens omkostningselementer baseret på ankomst og afgangstider.

Felterne skal læses som værdier i prioriteret rækkefølge:

< < < < < < < <

Ydermere er det en nødvendighed, at omkostningselementerne er eksponentielt stigende, for at vi er sikret en optimal interaktion delprogrammerne imellem.

Den måske største kritik af en LIFO-omkostning er håndteringen af indkomne togkompositioner. Ved ikke yderligere at tage højde for disse situationer, vil vi risikere at opsplitte indkomne togkompositioner og således parkere disse på forskellige spor. Omvendt kan vi godt betragte en togkomposition som én enhed i parkeringen, og som to enheder i hægtningen. Således kan vi få programmet til at sætte dem ved siden af hinanden i parkeringen, dog med risiko for en forringelse af løsningssuccesen.

4.2.3 Gennemgang af objektfunktionen

Vi har allerede påpeget, at vi ønsker at skabe interaktion mellem hægtningen og parkeringen, hvilket vi sikre gennem LIFO matricen. Vi ønsker ligeledes at kunne varetage hægtningen (og parkeringen) i

de tilfælde af, at der ikke er tilstrækkelig korrekt materiel, hvilket vi skal se lidt nærmere på i dette afsnit.

Til at starte med skal vi være bekendt med den hierarkiske opstilling af vores målsætning i hægtningen, således at vi kan fastsætte vores parameterværdier korrekt. Dette giver følgende hierarki:

1) Dække efterspørgslen af et togløb med mindst et togsæt
2) Lave hægtningen, der sikrer en stor interaktion med parkeringen
3) Efterleve så meget af efterspørgslen som muligt, givet muligt materiel

Herved sikrer DSB S-tog 1) at der kører togsæt på alle linjer, 2) at der findes en mulig parkering til dagens hægtning, og 3) at de forsøger at efterleve kundeefterspørgslen mest muligt og derved minimerer klager.

Punkt 1) sættes som et skarpt krav i restriktionerne, da vi skal dække efterspørgslen. Punkt 2) opfyldes ud fra LIFO matricen, som tidligere er beskrevet. Punkt 3) implementeres efter ønske fra DSB, hvor det skal være muligt at finde en løsning, selv om efterspørgslen ikke tilgodeses. Vi skal bruge informationen omkring togtypernes kapacitet, hvor et LHF udbyder 150 siddepladser, mens LHB har 366. Vi står således i en situation, hvor vi enten kan have for mange LHF vogne til LHB-efterspørgsler eller omvendt. Dvs. hhv. en underkapacitet eller en overkapacitet. Vi ønsker en ”ligevægt” mellem udbud og efterspørgsel, hvilket kan gøres med to justeringsparametre Y og Z.. Dette bringer os til følgende ”restriktion”:

Udbud – nedskrivning (Z) = efterspørgsel – nedskrivning (Y)

Vi ønsker, at skabe en hægtning, der efterlever mest muligt af efterspørgslen, hvorfor parametrene Z og Y skal påvirke objektfunktionen. Værdierne skal være så tilpas ”dyre”, at justeringen kun vælges, når det disponible materiel ikke er tilstrækkeligt. Herved bevares den hierarkiske struktur, som vi har opstillet. Restriktionen beskrives yderligere i Eq 4.1.4.

Vi har valgt at et ”point” i nedskrivningen svarer til én omkostning af den værst tænkelige LIFO tildeling. Dette medfører i praksis i modellen, at hvis en LHF enhed skal varetage en LHB efterspørgsel, så skal vi benytte 366-150=216 point for at skabe balance mellem udbud og efterspørgsel. Opstillingen af pointsystemet sikrer en korrekt fremgangsmåde for søgealgoritmen.

Ydermere vil vi til enhver tid anvende 2 LHF enheder, hvis disse er til rådighed, til at substituere en LHB. Vores objektfunktion kan nu beskrives som:

$$\min\left(\underbrace{\sum_{i\in I}\sum_{j\in J} x_{ij}^{m}\times l_{ij}}_{\substack{Omk\ ved\\ LIFO\ problematik}} + \underbrace{\max l_{ij}\times\sum_{i\in I} y_i}_{\substack{Strafomk\ ved\ under-\\ dækning\ af\ efterspørgsel}} + \underbrace{\max l_{ij}\times\sum_{i\in I} z_i}_{\substack{Strafomk\ ved\ over-\\ dækning\ af\ efterspørgsel}}\right) \qquad \text{Eq 4.1.1}$$

Strafomkostningen er sat til den største værdi i LIFO matricen.

Vi er nu klar til at se nærmere på den matematiske model for hægtningsproblemet.

4.2.4 Modelopbygning

Beslutningsvariable

Helt grundlæggende kan initial modellen beskrives ud fra vores beslutningsvariable, som benyttes til at beskrive hele systemet. Disse beskrives som:

$$x_{ij}^{m} = \begin{cases} 1\ hvis\ tognr.\ i \in I\ varetages\ af\ litra\ j \in J \\ 0\ ellers \end{cases}$$

Yderligere vil vi se på to beslutningsvariable defineret ved:

$$y_i \in \mathbb{Z}^+$$

$$z_i \in \mathbb{Z}^+$$

x_{ij}^{m} er binær og y_i og z_i er, som det forekommer, positive heltal. x_{ij}^{m} er knyttet til det grundlæggende hægtningsproblem, hvor "i" beskriver rækken af tilgængelige tognumre og "j" beskriver rækken af togsæt.

Objektfunktion

I 4.2.3 så vi hvilke forudsætninger, der er fastlagt omkring objektfunktionen og den hierarkiske opbygning, hvilket resulterede i følgende objektfunktion:

$$\min\left(\underbrace{\sum_{i\in I}\sum_{j\in J} x_{ij}^{m} \times l_{ij}}_{\substack{Omk\ ved\\ LIFO\ problematik}} + \underbrace{\max l_{ij} \times \sum_{i\in I} y_i}_{\substack{Straf omk\ ved\ under-\\ dækning\ af\ efterspørgsel}} + \underbrace{\max l_{ij} \times \sum_{i\in I} z_i}_{\substack{Straf omk\ ved\ over-\\ dækning\ af\ efterspørgsel}}\right) \qquad \text{Eq 4.1.1}$$

I det efterfølgende ser vi på de restriktioner, som er knyttet til hægtningen.

Restriktioner

Fra det overordnede assignmentproblem[11] optræder der to grundlæggende restriktioner. Disse kommer i vores model til udtryk ved, at hvert togsæt makismalt må hægtes en gang, og hvert tognummer skal varetages af minimum et togsæt.

$$\sum_{i\in I} x_{ij}^{m} \leq 1 \qquad \forall\, j \in J \qquad \text{Eq 4.1.2}$$

$$\sum_{j\in J} x_{ij}^{m} \geq 1 \qquad \forall\, i \in I \qquad \text{Eq 4.1.3}$$

I tilfælde af, at der ikke er tilstrækkeligt materiel til at varetage antallet af påkrævede tognumre, vil der forekomme en supplering af ekstra togsæt. Disse positioneres som regel fra HTÅ.

Efterspørgselsrestriktionen er allerede blevet nævnt under objektfunktionen på grund af dens indflydelse heri. Gennem restriktionen ønsker vi at tilfredsstille efterspørgslen d_i, såfremt det er muligt, og i tilfælde af at d_i ikke kan tilgodeses, skal der reageres ud fra to grundlæggende scenarier.

1) ved at indsætte overkapacitet af togsæder, beskrevet med en variabel z_i
2) ved at indsætte underkapacitet af togsæder, beskrevet med en variabel y_i

Vi kan nu bruge y_i og z_i til at justere vores restriktion, hvormed vi kan lave en mulig hægtning. Hvis vi definerer udbuddet fra hvert togsæt s_j og efterspørgslen d_i kan vi opstille restriktionen:

$$\sum_{j\in J} x_{ij}^{m} * s_j - z_i = d_i - y_i \qquad \forall\, i \in I \qquad \text{Eq 4.1.4}$$

eller i uddybet form

[11] Assignment Problems, kapitel 4, s. 73

$$\sum_{j \in J} \underbrace{x_{ij}^{m} * s_j}_{\substack{Udbudte \\ mængde}} - \underbrace{z_i}_{\substack{nedskrivning \\ ved\ overkap.}} = \underbrace{d_i}_{\substack{eftersp. \\ mængde}} - \underbrace{y_i}_{\substack{nedskrivning \\ ved\ underkap.}} \qquad \forall\, i \in I \qquad \text{Eq 4.1.4}$$

Endvidere er der en maksimal grænse for togkompositioner, som vi omtalte i afsnit 3.5.1. Dette bidrager til følgende restriktion:

$$\sum_{j \in J} x_{ij}^{m} \leq 2 \qquad \forall\, i \in I \qquad \text{Eq 4.1.5}$$

Ovenstående restriktioner kan defineres som primære infrastrukturelle begrænsninger og fysiske rammer for de togsæt, der varetager togtrafikken. Ud fra disse krav og objektfunktion kan vi i den helt simple grad hægte den virtuelle plan over i en faktisk udrulning. Men typisk er hægtningen mere kompleks og afhængig af flere enhedsspecifikke faktorer.

Vi har tidligere beskrevet sikkerhedskrav for serviceeftersyn som en af de større udfordringer, DSB S-tog står overfor. Helt grundlæggende gør det sig gældende, at togsæt ikke må overstige 50.000 km (50 Mm), før de skal være sendt til vedligeholdelse i HTÅ. Hvis togsættet ikke er omdirigeret i tide, betyder det, at DSB S-tog ikke kan benytte den enkelte enhed til driften. Herved vil en hægtning ikke længere være mulig. Vi har derfor behov for at definere en række variable, som indeholder den relevante information. Vi er nødsaget til at vide, hvor langt togsættet har kørt inden dagens hægtning, hvilket vi kan beskrive med α_j. Desuden skal vi vide, hvor lang distancen er, som ligger bag de enkelte tognumre, således at vi kan se, hvilke der er tilladte for enheden. Denne distance angives med δ_i. Dette kan skrives som:

$$\alpha_j + \sum_{i \in I} x_{ij}^{m} * \delta_i \leq 50Mm \qquad \forall\, j \in J \qquad \text{Eq 4.1.6}$$

- Dvs. α_j er antallet af kilometer, som togsæt j har tilbagelagt siden sidste eftersyn og δ_i er tognummer i's kilometerdistance.

I tråd med vedligeholdelsesrestriktionen findes også påfyldning af friktionssand, som sker hvert 10 Mm. Dette kan både foretages i UND og HTÅ, men eftersom det sker med en højere frekvens, er det under skarpt opsyn. Tilsvarende kan sandeftersyn skrives som:

$$\beta_j + \sum_{i \in I} x_{ij}^m * \delta_i \leq 10Mm \qquad \forall\, j \in J \qquad \text{Eq 4.1.7}$$

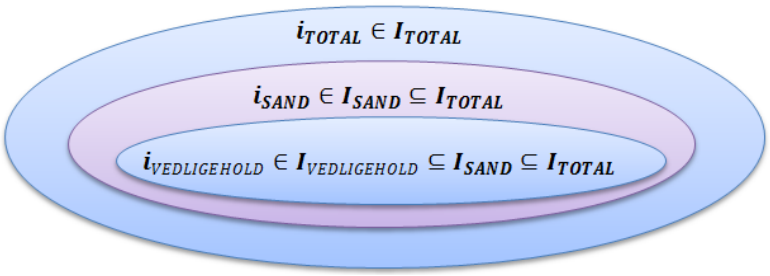
- Hvor β_j er antallet af kilometer, som togsæt j har tilbagelagt siden sidste sandpåfyldning.

Det er essentielt for DSB S-togs målsætning at være på forkant med disse restriktioner, når et togsæt nærmer sig sit vedligeholdelseseftersyn eller sin sandpåfyldning. Det betyder, at når et togsæt nærmer sig sine begrænsninger, skal modellen formå at omdirigere enheden tidsnok, så dette kan indgå i driften, som en indtægtsgivende positionering.

Dette tager vi hånd om med en løs fiksering af beslutningsvariablene. Dette gøres ved at sætte en ekstra bibetingelse ind på de variable, som angiver en mulig servicerute. Vi har derfor behov for at få præciseret variablen "*i*" for hvilken type togløb, der er tale om. Modellen skal således håndtere tre typer af togløb:

$$i_{SAND}, i_{VEDLIGEHOLD}, i_{REST} \in I_{TOTAL}$$

For at klarlægge hvor fikseringen er nødvendig, kan vi med fordel forsøge at illustrere vores delmængder i I. i_{TOTAL} udgør samtlige tognumre og kan opdeles i to undergrupper. Den ene kategori kalder vi for i_{REST} og den beskriver de tognumre, der ikke berøres af hverken sand eller vedligeholdelseseftersyn. Dette er enheder, der sendes ud på enten C, E, F eller H linjen. Den anden kategori kan opsplittes i $i_{VEDLIGEHOLD}$ og $i_{SAND.}$ $i_{VEDLIGEHOLD}$ er alle de tognumre, der kører på B og Bx linjen, hvor i_{SAND} både er B og Bx, samt A linjen. Herunder er der skitseret, hvilke *i* der er delmængde af den samlede portefølje over tognumre.



Figur 4.4 : Mængden I med tilhørende delmængder

Vi har nu præciseret de linjer, der står til rådighed for omdirigering, hvilket resulterer i to nye restriktioner for hhv. sand og vedligeholdelse. Hertil findes fire vigtige faktorer, som udspænder restriktionerne.

- Hvor langt har togsættet kørt til dags dato.
- Hvor langt må togsættet maksimalt køre inden vedligeholdelseseftersyn eller sandeftersyn.
- Distancen på den korteste rute i den kommende planlægning foruden de tognumre, der går til vedligeholdelse og sandpåfyldning, eks. $\min \delta_{i \in I \setminus \{I_{VEDLIGEHOLD}\}}$
- Distancen på servicelinjen, som måles i km fra startstation til anden ankomst på servicedepotet. eks. $\max \delta_{i \in I_{VEDLIGEHOLD}}$

Et togsæt skal omdirigeres i tilfælde af, at den længde, toget har kørt, samt distancen af den korteste rute overskrider den maksimalt tilladte distance fratrukket distancen for en given servicelinje. Eksempelvis kan scenariet forekomme:

- Hvor et togsæt har kørt 49.700 km og den kortest mulige distance for togløb $\in \mathrm{I} \setminus \{\mathrm{I}_{\mathrm{VEDL}}\}$ er 200 km. Samtidig er serviceruten 134 km og maksimalt må enheden køre 50.000 km inden syn. Dette togsæt omdirigeres nu til B el. Bx, da det ikke er i stand til at varetage det korteste togløb i netværket samt tognummeret til HTÅ.

Dette kan matematisk formuleres som:

$$\sum_{i \in I_{VEDL}} x_{ij}^{m} \geq f_j \quad hvor\ f_j = \begin{cases} 1\ hvis\ \alpha_j + \min \delta_{i \in I \setminus \{I_{VEDL}\}} \geq 50Mm - \max \delta_{i \in I_{VEDL}} \\ 0\ ellers \end{cases} \quad \forall\, j \in J \qquad \text{Eq 4.1.8}$$

f_j er en binær variabel, som "tændes" i tilfælde af, at togsættet skal sendes til vedligeholdelse. Derfor er vi nødsaget til at omdirigere det pågældende togsæt over til servicelinjen, der kører mod HTÅ.

Ligeledes kan restriktionen for sandpåfyldning formuleres:

$$\sum_{i \in I_{SAND}} x_{ij}^{m} \geq g_j \quad hvor\ g_j = \begin{cases} 1\ hvis\ \beta_j + \min \delta_{i \in I \setminus \{I_{SAND}\}} \geq 10Mm - \max \delta_{i \in I_{SAND}} \\ 0\ ellers \end{cases} \quad \forall\, j \in J \qquad \text{Eq 4.1.9}$$

g_j er en binær variabel, som "tændes" i tilfælde af, at togsættet skal sendes til sandeftersyn. Derfor er vi nødsaget til at omdirigere det pågældende togsæt over til servicelinjen, der kører mod HTÅ eller UND.

Afslutningsvis defineres restriktionen for beslutningsvariablene:

$$x_{ij}^{m} \in \{0,1\} \quad \forall\, i \in I \wedge j \in J$$

$$y_i \in \mathbb{Z}^+ \quad \forall\, i \in I$$

$$z_i \in \mathbb{Z}^+ \quad \forall\, i \in I$$

4.2.5 Matematisk model

Modellens struktur og opbygning er nu beskrevet ovenfor, og her er blot en opsamling af den samlede matematiske model for hægtning mellem togsæt og tognumre for DSB S-tog.

$$\min\left(\sum_{i\in I_{TOTAL}}\sum_{j\in J} x_{ij}^{m} \times l_{ij} + \max l_{ij} \times \sum_{i\in I_{TOTAL}} y_i + \max l_{ij} \times \sum_{i\in I_{TOTAL}} z_i\right) \qquad \text{Eq 4.1.1}$$

$s.t.$

$$\sum_{i\in I_{TOTAL}} x_{ij}^{m} \leq 1 \qquad \forall\, j \in J \qquad \text{Eq 4.1.2}$$

$$\sum_{j\in J} x_{ij}^{m} \geq 1 \qquad \forall\, i \in I_{TOTAL} \qquad \text{Eq 4.1.3}$$

$$\sum_{j\in J} x_{ij}^{m} * s_j - z_i = d_j - y_i \qquad \forall\, i \in I_{TOTAL} \qquad \text{Eq 4.1.4}$$

$$\sum_{j\in J} x_{ij}^{m} \leq 2 \qquad \forall\, i \in I_{TOTAL} \qquad \text{Eq 4.1.5}$$

$$\alpha_j + \sum_{i\in I_{TOTAL}} x_{ij}^{m} * \delta_i \leq 50Mm \qquad \forall\, j \in J \qquad \text{Eq 4.1.6}$$

$$\beta_j + \sum_{i\in I_{TOTAL}} x_{ij}^{m} * \delta_i \leq 10Mm \qquad \forall\, j \in J \qquad \text{Eq 4.1.7}$$

$$\sum_{i\in I_{VEDL}} x_{ij}^{m} \geq f_j \quad hvor\ f_j \qquad \text{Eq 4.1.8}$$

$$= \begin{cases} 1\ hvis\ \alpha_j + \min \delta_{i\in I \setminus \{I_{VEDL}\}} \geq 50Mm - \max \delta_{i\in I_{VEDL}} \\ 0\ ellers \end{cases} \qquad \forall\, j \in J$$

$$\sum_{i \in I_{SAND}} x_{ij}^{m} \geq g_j \quad hvor\ g_j$$

$$= \begin{cases} 1\ hvis\ \beta_j + \min \delta_{i \in I \setminus \{I_{SAND}\}} \geq 10Mm - \max \delta_{i \in I_{SAND}} \\ 0\ ellers \end{cases} \quad \forall\ j \in J \qquad \text{Eq 4.1.9}$$

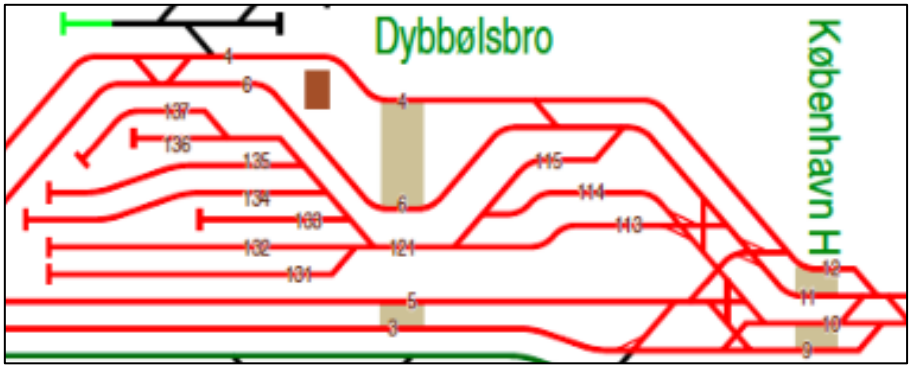
$$x_{ij}^{m} \in \{0,1\} \quad \forall\ i \in I \wedge j\ \in J$$

$$y_i, z_i \in \mathbb{Z}^{+} \quad \forall\ i \in I$$

Hvor x_{ij}^{m} er binære beslutningsvariable, som viser hægtning mellem togsæt j og tognummer i, y_i og z_i er positive heltal og viser op- og nedjusteringer af udbud-efterspørgselsrestriktionen, s er udbuddet fra det pågældende togsæt, d er efterspørgslen ved det pågældende tognummer, α_j er antal kørte km siden sidste eftersyn, δ_i er længden af tognummer i, β_j er antal kørte km siden sidste sandpåfyldning og sidst er f_j og g_j binære variable som fikserer en linje til vedligeholdelse eller sandpåfyldning, set som en ja- nej-variabel.

4.3 Parkeringsproblemet

I 4.2.2 blev vi introduceret til de fysiske rammer, som finder sted på KH, hvor vi vil koncentrere os om de 7 LIFO spor, der er tilgængelig på depot Vest. Placeringerne og infrastrukturen ses på figur:



Figur 4.5 : infrastruktur for depot Vest på KH, som består af depotspor 131-137.

Som det fremgår på figur 4.5, opstår der en naturlig begrænsning fra infrastrukturen ifm. hvor mange parkeringer, der kan forekomme samtidigt. Teoretisk vil man kunne parkere 7 togsæt på 7 depotspor

samtidigt, men da der kun optræder et tilløbsstykke, spor 121, til depotområdet, vil det i praksis ikke være muligt. Ligeledes vil depotbemandingen resultere i naturlige begrænsninger i netværket. Planlagt er der 4-5 køremænd (*KMD*), hvilket giver et maksimum på tilsvarende deponeringer. Begge forhold er der taget hånd om i den langsigtede planlægningsfase, hvorfor vi ikke inkluderer dette aspekt i modellen.

DSB S-tog har et ønske om at fastsætte en begrænsning på det mellemliggende tidsinterval mellem to parkerede togsæt. Dette sikrer, at en KMD har tid til at hente og bringe togsæt i løbet af driften, således at to KMD ikke blokerer for hinanden på samme depot. Tidsintervallet skaber samtidig en langt mere robust parkering, som kan håndtere mindre uforudsete forstyrrelser i netværket. Derfor indlægges et sikkerhedsinterval på 10 minutter mellem hver parkering.

Den nuværende arbejdsopgave for parkeringen løses delvist i cirkulationsplanlægningen og delvist i dag-til-dag planlægningen. I cirkulationsplanlægningen foretages parkeringerne af de virtuelle tognumre. I dag-til-dag planlægningen er det operatørens opgave at få tildelt de ankomne togsæt til depotpladserne, således at det enkelte togsæt kan varetage det togløb, som skal afgå fra denne plads. I den manuelle proces foretages således selve hægtningen i samme fase, men medfølger, at parkeringen er relativt låst.

Eftersom vi allerede har foretaget en hægtning i første del af modellen, bliver depotplanlægningen i cirkulationsplanlægningen overflødig. Vi har således ikke på forhånd besluttet, hvilke depotpladser der har specifikke afgange (tognumre), men vælger at placere tognumre og togsæt simultant.

Vi har valgt at ændre denne fremgangsmetode, da vi mener, at planlægningsfasen i cirkulationsplanlægningen bliver overflødig.

De tidligere omtalte hægtninger vil blive refereret til som deponeringer. Her kan der være tale om enkelte hægtninger mellem et togsæt og et tognummer, men også tilfælde af en togkomposition, der er tildelt flere tognumre.

4.3.1 Modificering/klargøring af data

Det kan være problematisk at definere, hvordan en optimal løsning til et parkeringsproblem udformes. Set fra DSB S-togs synsvinkel vil en optimal løsning være tilstrækkelig, såfremt der er en mulig løsning, hvor alle restriktioner blive overholdt.

Under hægtningsproblemet så vi tidligere fordelen ved at opdele tognummeret, så det blev synliggjort, hvorvidt et tognummer indgik i flere togløb. Her lå fokus på de eventuelt forskellige kilometerrestriktioner, der er pålagt hvert togløb. Dette er ikke længere relevant, og derfor kan vi med fordel lade hvert tognummer optræde én gang. Til tognummeret skal der være tilknyttet afgangs- og ankomsttid samt den efterspurgte depotkapacitet. En kort specifikation af påkrævet information ses herunder:

Tognr	Type1	Type2	Afgang	Tog1	Tog2	Kap	MIN_ankomst	MAX_ankomst
32219	LHB		06:28:00	Togsæt 28		366	20:50:00	20:50:00
52117	LHB	LHF	05:41:00	Togsæt 16	Togsæt 29	516	19:27:00	19:45:00

Tabel 4.2 : Output omdannet fra hægtningsproblemet, hvor hvert tognummer kun optræder én gang.

Vi kan se ud fra tabel 4.2, at hver tognummer blot optræder en gang men med en samlet kapacitet for antal togsæt, som varetager togløbet. Vi ønsker derved at parkere udgående togsæt på samme depot, hvorfor vi skal kende den tidligste og seneste ankomst for togsættene, så vi kan sikre os, det omtalte tidsinterval på 10 minutter mellem parkeringen af tognumrene.

Tabel 4.3 viser sporlængden på de 7 depotspor og vil blive benyttet til at sikre, at vi ikke parkerer mere materiel end depotets kapacitet tillader.

Depotspor	131	132	133	134	135	136	137
Sporlængde (m)	512	509	187	425,7	336	171,5	260

Tabel 4.3 : Depotlængder beskrevet i meter på Depot Vest KH.

Vi kan nu påbegynde opsætningen af den matematiske model for parkeringsproblemet.

4.3.2 Modelopbygning

Parkeringsproblemet udspændes af i alt 53 virtuelle depotpladser[12], som defineres fra *i* = [1,2,..., 53], og et varierende sæt af hægtninger mellem togsæt og tognumre, som henvises til de enkelte deponeringer og betegnes med *j*. Afhængigt af resultatet ved hægtningen kan antallet af deponeringer variere alt afhængigt af, hvilke hægtninger der bliver betragtet individuelt eller sammensat. Depotpladserne er fordelt på 7 depotspor, som angives d = [1,2, ... ,7].

Beslutningsvariable:

Vi definerer beslutningsvariablene ud fra, hvilken depotplads en given hægtning tildeles, hvilket kan beskrives som:

$$x_{ij}^{p} = \begin{cases} 1 \; hvis \; plads \; i \; bliver \; benyttet \; af \; deponering \; j \\ 0 \; ellers \end{cases}$$

Det fremgår af x_{ij}^{p} er beslutningsvariablene binære.

Objektfunktion

Vi sigter på at maksimere antallet af parkeringer, så vi sikrer at flest mulige togsæt bliver parkeret på depotsporene.

$$\max \sum_{i \in I} \sum_{j \in J} x_{ij}^{p} \qquad \text{Eq 4.2.1}$$

Restriktioner

For at en samlet parkering er mulig, skal alle togsæt parkeres, mens det sikres, at en depotplads maksimalt benyttes en gang:

$$\sum_{i \in I} x_{ij}^{p} = 1 \qquad \forall \, j \in J \qquad \text{Eq 4.2.2}$$

[12] En virtuel depotplads kan indeholde 1 type LHF enhed, samt godt ½ type LHB. En illustration af depotpladser på de enkelte depotspor kan se i bilag 3

$$\sum_{j \in J} x_{ij}^{p} \leq 1 \qquad \forall\, i \in I \qquad \text{Eq 4.2.3}$$

Kravene til parkeringsproblemet udspændes af flere specifikke restriktioner, som øger kompleksiteten og begrænser mulighedsområdet i den samlede løsning. Disse vil i det efterfølgende blive beskrevet.

Hvert depotspor har sine kapacitetsbegrænsninger, og vi er derfor nødt til at sikre os, at længden af depotsporet ikke overskrides. Her er det vigtigt, at vi tager højde for togkompositioner og deres øgede kapacitetsforbrug. Dette kan formuleres som:

$$\sum_{j \in J} \sum_{i \in I_d} x_{ij}^{p} * k_j \leq K_d \qquad \forall\, d \in D \qquad \text{Eq 4.2.4}$$

Vi ønsker at sikre at den samlede sum af tildelte togsæts længde, som optræder på et bestemt depot *d* defineret ved de $i'er \in I_d$, ikke overskrider den samlede sporlængde på det pågældende depot.

For at forstå at alle virtuelle pladser på depotet ikke skal benyttes, kan vi opstille et lille eksempel. Vi så på tabel 4.3, at sporlængden på depot 133 er 187 m. Herved er der plads til $\frac{187}{42{,}58} \approx 4LHF$. Reelt set giver det 4 virtuelle pladser. Hvis depotsporet nu fyldes med 2 LHB enheder, vil der ikke længere være plads til yderligere parkeringer. Således fyldes sporet hvor vi reelt set har 2 virtuelle pladser tilovers, men den samlede kapacitet K_d er opbrugt.

For at driften kan afvikles uden krydsninger, er det et krav, at placeringen af de udgående togsæt står i stigende rækkefølge i forhold til deres afgangstid. Da vi arbejder med LIFO-spor, skal vi sikre, at det togsæt, som har tidligst afgang, ligeledes står forrest parkeret på depotsporet. Der fastsættes yderligere et tidsinterval imellem afgange for to parkeringer på samme depot. Dette minimum beskriver antallet af minutter, som skal sikre, at køremændene (*KMD*) på depotområdet kan nå at sætte enhederne i drift. Dette tidsrum betegnes som t_{afg}. Restriktionen kan nu formes som:

$$\sum_{j \in J} x_{ij}^{p} * \left(af_j - t_{afg}\right) \geq \sum_{j \in J} x_{i+1,j}^{p} * af_j \qquad \forall\, i \in I_d \backslash \{\max(i_d)\}, \qquad d \in D \qquad \text{Eq 4.2.5}$$

Restriktionen angiver nu, at den enhed, der bliver sat på den første plads på depotsporet, har en afgangstid, der fratrukket det nødvendige tidsinterval, er større eller lig med afgangstiden for tognummeret, der kan placeres på den efterfølgende depotplads etc. Dog er det vigtigt at notere sig, at det aldrig omfatter den yderste plads på depotet, eftersom $\max(i_d) \in I_d$ og $\min(i_{d+1}) \in I_{d+1}$ikke afhænger af hinanden.

Kompleksiteten i problemet stiger, når vi ønsker en løsning, der både tager højde for tildelingen på depotsporene i stigende orden over afgangstider for hvert tognummer, samt en faldende orden for de enkelte ankomsttider for de tilknyttede togsæt. Vi har tidligere defineret, at de yderst placerede togsæt på sporet skal være de senest ankomne, hvorfor vi tidligere så på krav omkring faldende ankomsttider.

For at få ankomsttider og afgangstider til at interagere med hinanden, kan vi derfor med fordel bruge den inverse addition af ankomsttiden[13]. Herved skaber vi en fordelagtig situation, hvor vi har et krav om, at både afgang og den inverse addition af ankomsttider skal være stigende desto længere inde på depotsporet, vi kommer.

Med de nye ankomsttider er det nu muligt tilsvarende at opstille et stigende krav for disse. Her skal vi tage højde for sammensatte enheder, hvilket afspejles i en kontrol af den største og mindste ankomsttid. Ydermere ses der på kravet om et minimalt tidsinterval, t_{ank}, som er nødvendigt mellem to efterfølgende parkeringer. Restriktionen kan opstilles på følgende vis:

$$\sum_{j \in J} x_{ij}^{p} * (\min\_an_j - t_{ank}) \geq \sum_{j \in J} x_{i+1,j}^{p} * \max\_an_j \qquad \forall\, i \in I_d \setminus \{\max(i_d)\}, \quad d \in D \qquad \text{Eq 4.2.6}$$

For ikke sammensatte togenheder er *min_an*$_j$ og *max_an*$_j$ identisk og sikrer, at rækkefølgen overholdes. I forbindelse med togkompositioner er vi nødsaget til at sætte et krav om, at det sidst ankomne togsæts ankomsttid stadigvæk er større eller lig ankomsttiden for det togsæt, der efterfølgende skal parkeres på næste depotplads. Dertil skal der ligeledes indlægges et tidsinterval,

[13] Den inverse additions er den værdi af x, der sikre, at a+x=0. Ved et 24-timers ur ønskes derfor $x + a_{ank} = 24$. Eksempelvis vil en ankomsttid 20:30:00 have den inverse additions ankomsttid 03:30:00.

som sikrer, at KMD kan varetage parkeringen. Den sidste plads på depotet, $\max(i_d)$, skal ikke indgå i restriktionen, da den ikke påvirkes af den efterfølgende plads, idet denne plads tilhører et nyt depot.

Afslutningsvis sættes et krav om, at vores beslutningsvariable x_{ij} er binære:

$$x_{ij}^{p} \in \{0,1\} \quad \forall\, i \in I \wedge j \in J$$

4.3.3 Matematisk model

Parkeringsproblemets struktur og opbygning er beskrevet ovenfor, og her ses en opsamling af den samlede matematiske model for deponering af hægtningerne fra tidligere.

$$\max \sum_{i \in I} \sum_{j \in J} x_{ij}^{p} \qquad \text{Eq 4.2.1}$$

$s.t.$

$$\sum_{i \in I} x_{ij}^{p} = 1 \qquad \forall\, j \in J \qquad \text{Eq 4.2.2}$$

$$\sum_{j \in J} x_{ij}^{p} \leq 1 \qquad \forall\, i \in I \qquad \text{Eq 4.2.3}$$

$$\sum_{j \in J} \sum_{i \in I_d} x_{ij}^{p} * k_j \leq K_d \qquad \forall\, d \in D \qquad \text{Eq 4.2.4}$$

$$\sum_{j \in J} x_{ij}^{p} * \left(af_j - t_{afg}\right) \geq \sum_{j \in J} x_{i+1,j}^{p} * af_j \qquad \forall\, i \in I_d \setminus \{\max(i_d)\}, \quad d \in D \qquad \text{Eq 4.2.5}$$

$$\sum_{j \in J} x_{ij}^{p} * (\min\_an_j - t_{ank}) \geq \sum_{j \in J} x_{i+1,j}^{p} * \max\_an_j \qquad \forall\, i \in I_d \setminus \{\max(i_d)\}, \quad d \in D \qquad \text{Eq 4.2.6}$$

$$x_{ij}^{p} \in \{0,1\} \quad \forall\, i \in I \wedge j \in J$$

Hvor x^p er binære beslutningsvariablene, k er kapaciteten en given deponering kræver, K_d er den samlede kapacitet, der er på et givent depotspor, af er afgangstider for tognumre, t_{afg} er tidsintervallet mellem tilladt parkering for tognumre, min_an er den mindste "justerede" ankomsttid i en given deponering, max_an er den seneste "justerede" ankomsttid og t_{ank} er tidsintervallet mellem en tilladt parkering af togsæt.

5. Løsning af BSS

5.1 Test af modelopsætning

Vi vil i dette afsnit undersøge hvilken fremgangsmåde, der skaber de bedste forudsætninger for at finde en løsning til problemet, hvor parkeringen af togsættene bliver mulig. Løsningen til hægtningsproblemet vil mere eller mindre altid være gennemførlig, eftersom vi antager, at der er enheder nok til at dække de efterspurgte togløb. Derfor vil vi fastholde en løsning af hægtningen og herfra tage udgangspunkt i den opstillede model i parkeringsproblemet. Her vil vi foretage justeringer i de tilhørende restriktioner for at se, om det er muligt at ændre på dele af modellen for at kunne sikre en bedre, hurtigere og mere robust parkering. Samtidig vil vi benytte de bagvedliggende funktioner i SAS til at styre, hvorledes søgealgoritmen skal tilgå modellen for at effektivisere beregningstiden for løsningen.

Herunder gennemgår vi fire modeller, som vi ønsker at teste. Under hver model vil vi begrunde de tanker, som ligger til grund for ændringerne.

Model 1

Her tager vi udgangspunkt i den allerede beskrevne model af parkeringsproblemet. Vi ønsker her at finde en løsning, der parkerer alle togsæt. Vi har en situation, hvor antallet af togsæt/togkompositioner svarer til den nedre grænse for antallet af parkeringer, som vil resultere i en mulig parkering.

$$\max \sum_{i \in I} \sum_{j \in J} x_{ij}$$

$$ubb.$$

$$\sum_{i \in I} x_{ij} = 1 \qquad \forall\, j \in J$$

Denne tilgang accepterer kun en løsning, hvor antallet af togsæt/togkompositioner skal svare til antallet af parkeringer. Løsningen vil derfor ikke være gennemførlig, såfremt der ikke kan findes en fuldstændig løsning til parkeringen.

Da vi ønsker at danne et beslutningsstøttesystem, vil det være en begrænsende faktor, at modellen ikke vil præsentere brugeren for en ikke-fuldkommen løsning.

Model 2

Vi vælger nu at opstille en model, hvor vi lemper på restriktionen, så det ikke længere er et krav, at alle hægtninger skal besætte en depotplads. Vi ønsker her at parkere flest mulige togsæt, hvilket betyder, at der kan fremkomme løsninger, hvor vi ikke opnår en fuldstændig parkering. Vi ser her en fordel i, at operatøren vil blive fremlagt et parkeringsforslag selv i de situationer, hvor en fuldstændig løsning eventuelt ikke findes. Vi holder således fast i at maksimere antallet af parkeringer, mens vi relaxerer restriktionen, der tidligere krævede, at alle togsæt til deponering skulle parkeres. Dette kommer til udtryk ved en ændring af Eq 4.2.2, hvor der ikke længere er et krav til parkering:

$$\sum_{i \in I} x_{ij} \leq 1 \qquad \forall\, j \in J$$

På denne måde øger vi sandsynligheden for at frembringe et resultat, da programmet accepterer en delvis løsning. Fordelen ved denne model er yderligere, at når vi kan acceptere delvise løsninger, kan vi opstille en nedre grænse (LB), som hurtigt og effektivt kan udelukke dele af mulighedsområdet ved beskæringer i Branch and Bound-træet.

Model 3

Med udgangspunkt i model 1 tilføjer vi nu yderligere en restriktion. Når togsæt/togkompositioner er blevet tildelt et tognummer under hægtningsprocessen, har vi i den tidligere opsætning taget højde for, at en udgående togkomposition bestående af to togsæt med forskellige ankomsttider vil blive parkeret på samme depotspor efter hinanden. Vi har som udgangspunkt valgt ikke at koncentrere os omkring split af indkommende togkompositioner under hægtningen, og vi vil med denne tilføjelse forsøge at sikre at sammenholde togkompositioner i de tilfælde, hvor det kan lade sig gøre.

Tidligere har vi ikke tilladt parkering af en togkomposition medmindre, at de enkelte togsæt i togkompositionen under hægtning er blevet tildelt samme afgangstid/togløb. Dette skyldes kravet til et 10 minutters interval mellem hvert parkeret togsæt på et depotspor. I model 3 vil vi teste resultatet af at se bort fra dette krav, så ankomne togkompositioner, såfremt hægtningen tillader det, parkeres på samme depotspor.

Dette kræver endnu en restriktion, som tilnærmelsesvis minder om Eq 4.2.6. Her benyttes informationen omkring den mindste og største afgangstid, som sikrer, at der vil være et 10 minutters interval til både foranliggende og bagvedliggende togsæt. Restriktionen defineres som:

$$\sum_{j \in J} x_{ij} * \left(\min\_af_j - t_{afg}\right) \geq \sum_{j \in J} x_{i+1,j} * \max\_af_j \qquad \forall\, i \in I_d \setminus \{i_{max}\}, \qquad d \in D$$

Vi skal således også fastsætte en hierarkisk rækkefølge for, hvornår vi ønsker at se på togkompositionen. Det betyder, at en udgående togkomposition har en højere værdi med henblik på parkeringen i forhold til en indkommende togkomposition. Mao. i tilfælde af at et togsæt fra en indkommende togkomposition tildeles en ny afgående togkomposition, så favoriseres afgangen i parkeringen.

En ulempe ved denne tilgang kan være, at vi indskrænker mulighedsområdet, da der er færre mulige parkeringer. Men hvorvidt dette har en effekt på løsningen testes netop i denne tilgang. Vi tester primært på, om antallet af løsninger reduceres i forhold til tidligere, samt hvorvidt beregningstiden øges bemærkelsesværdigt. Modellen indeholder stadig de strenge krav fra model 1, som kræver, at alle togsæt parkeres for en gennemførlig løsning.

Model 4

I denne model benytter vi tankerne bag model 2 og 3, hvor vi tillader en delvis løsning, samtidigt med at vi tager højde for parkering af indkomne togkompositioner.

5.1.1 Baggrund for tests

Gennem udviklingen af programmet har vi brugt et empirisk eksempel, hvortil den praktiserede løsning hos MAS er kendt[14]. Vi benytter dette datagrundlag og danner herfra en hægtning, hvor togsættene nu skal parkeres. Når vi referer til muligheden omkring en løsning, vil der blive henvist til den testede model, og hvorvidt SAS er i stand til at finde en løsning ud fra de begrænsninger, vi sætter i algoritmen. Hvorvidt, programmet finder en løsning, afhænger bl.a. af en eventuel begrænsning på antallet af noder, som BnB-træet tillader at udføre eller en tidsbegrænsning. Når der skelnes mellem begrænsningen for de to parametre, skyldes det, at beregningstiden varierer pr. node alt afhængigt af, hvilken algoritme SAS er indstillet til. Nedenfor begrundes både valget for tidsforbrug og maksimalt nodevalg:

[14] Se bilag 2

Maksimalt tidsforbrug

Vi sætter en grænse på 10 minutter for selve kørslen af programmet. Tiden er sat ud fra en forventning omkring, hvilken beregningstid MAS vil acceptere. Ud fra erfaringer gennem et signifikant antal af testkørsler har det vist sig, at en manglende løsning efter længere beregningstid skyldes, at søgealgoritmen har valgt at bevæge sig ud af en gren, hvor der ikke findes nogen brugbar løsning. Det betyder, at antallet af beregninger kan stige ekstremt, før en given løsning er fundet. Det vil derfor kunne betale sig at starte programmet forfra.

Maksimalt nodevalg

Antallet af gennemløbede noder er sat til 100.000. Antallet svarer tidsmæssigt i omegnen til den allerede nævnte tidsbegrænsning og er sat ud fra selvsamme observationer. Ideen ved ikke at tillade programmet at køre i en uendelighed er, at vi kan gå ind og ændre på andre faktorer, hvis der ikke er fundet en løsning inden for det accepterede antal gennemløb eller tidsintervaller.

Fremgangsmåde

I afsnit 2.3 beskrev vi, hvilke fire metodevalg SAS udbyder, og hvilke forskellige søgealgoritmer, der benyttes gennem noderne. Vi vil nu benytte metoderne *Automatic, BestBound, BestEstimate og Depth,* samt andre hjælpestatements til at teste, hvilken effekt sammensætningen af disse har i forhold til at finde en løsning. Da testene forholder sig til et fast datagrundlag, er vi velvidende omkring muligheden for anderledes resultater ved anden empiri. I figur 5.1 ses kombinationsmulighederne for testundersøgelsen:

nodesel	*presolver* *primalin* *heuristics* *allcuts*
Automatic	
BestBound	Aggressive
BestEstimate	(*Automatic*)
Depth	

Figur 5.1 : viser forudsætninger for kombinationsmulighederne ved testkørslerne.

Figur 5.1 viser, hvorledes vi konstant ved testene vil fastholde en af de fire nodevalg. Ved hver hovedtest varierer vi efterfølgende på de fire hjælpestatements *presolver, primalin, heuristics* og *allcuts*, hvor vi samtidig undersøger tilgangen *aggressive* eller *automatic*. Vi tester hermed op imod tilgangen ved default og aggressive for at se hvilken indvirkning, de har på algoritmen. For nærmere beskrivelse af tilvalg se afsnit 2.3.

5.1.2 Diskussion

Opstillingen af de fire modeller har til formål at skabe et billede af hvilken model, der sikrer de bedste forudsætninger for en brugbar løsning til parkeringsproblemet. Yderligere søger vi den tilgang, som skaber de bedste forudsætninger for en succesfuld løsning, hvor vi ønsker at minimere beregningstiden.

Resultatet af output kan komprimeres ned til følgende fire oversigtstabeller:

Testtype	*cpu time*	*nodes*	*Solution*
Ingen tilvalg	65	12.052	75%
Allcuts=aggresive	78	14.710	65%
Heuristik=aggressive	53	12.212	85%
Presolver=aggressive	61	21.542	70%
Primalin	64	12.025	75%
Alle tilvalg	79	10.559	55%

Approach	*cpu time*	*nodes*	*Solution*
Model 1	63	14.939	71%
Approach2	64	21.244	42%
Model 2	62	8.532	83%
Model 3	127	26.497	75%
Model 4	16	1.156	83%

Nodesel	*cpu time*	*nodes*	*Solution*
Automatic	70	12174	80%
BestBound	105	18031	70%
BestEstimate	45	11719	80%
Depth	46	13478	53%

Bedste løsning	*cpu time* noder*		*Solution*
Middel	9,25	385	100%

Tabel 5.1 : Her ses løsningerne fra diverse strategiske metodevalg, hjælpestatements, samt modeltilgange. "CPU time" er tiden angivet i sek., som det i gennemsnit har taget at komme frem til en fuldstændig løsning. Dvs. at alle løsninger som ikke var gennemførlige er frasorteret. Ligeledes beskriver "nodes" det antal noder, som i gns. er brugt på succesfulde løsninger Til sidst angives den succesrate, som modellen har tilvejebragt.

Det samlede resultat af alle testkørsler på modellerne kan ses i bilag 4.

Det gennemsnitlige tidsforbrug samt antallet af noder er kun baseret på kørsler, hvortil der er fundet en løsning. Valget er foretaget for at danne et mere retvisende billede af de enkelt modeller overfor hinanden. Givet at alle observationerne var medtaget, ville resultaterne bl.a. kunne afhænge af definitionen af den øvre grænse, hvilket vi mener vil være misvisende. Jf. bilag 4 ser vi, at der i tilfælde af ikke-komplette løsninger fremstår en høj behandlingstid, hvilket derfor har en stor betydning for den gennemsnitlige tid.

Vores valg af model afspejles primært gennem sandsynligheden for en mulig parkering og sekundært af beregningstiden ved løsningen, hvor antallet af noder er ubetydeligt. Tabel 5.1 viser, at der som

udgangspunkt findes en god løsning ved brug af Model 2 og 4, hvor vi inddrager følgende statements: ”heuristic = aggressivt” og ”nodesel = BestEstimate”. Model 1 og 2 minder til dels om hinanden, men er interessante at sammenligne pga. deres beregningstider. Fra resultaterne ses det, at løsningsgraden er bedre for model 2, hvilket kan tyde på, at den ved hjælp af den øvre grænse lettere kan frasortere overflødige dele af mulighedsområdet, således at programmet finder en løsning. Omvendt ser vi, at model 1 til tider præsenterer en løsning ekstremt hurtigt. I sidste instans er den gennemsnitlige ventetid for en brugbar løsning omtrent den samme.

Model 3 og 4 er resultatet af et forsøg på at behandle indkommende togkompositioner som samlede deponeringer. Umiddelbart ser vi ikke en forringelse af løsningen, og det kan skyldes, at fra hægtningsproblemet vil tildelingen af afgangstiderne til togsættene i togkompositionen oftest ligge relativt tæt på hinanden. Vi observerer derfor ofte et forbedrende resultat ved at sammensætte togkompositionerne. Vi kan tilsvarende komme i situationer, hvor hvert togsæt i kompositionen er hægtet til to togløb, der tidsmæssigt ligger langt fra hinanden. Dette lukker et større tidsinterval på depotsporet, og kan i nogle tilfælde forringe mulighedsområdet. Ligeledes kan vi komme i situationer, hvor de afgående tider er så tæt på hinanden, at KMD kan være nødsaget til at køre begge vogne ud på en gang.

Vi har hermed observeret en forbedring af vores basismodel og vil fremadrettet foretage vores gennemkørsler på baggrund af model 4. Samtidigt fastsætter vi i programmet en *”aggressiv”* søgeheurestik, hvor valget af forbedring i træet sker på baggrund af *”BestEstimate”*.

5.2 Præsentation af løsningen

Prototypen, som præsenteres i dette projekt, er blevet anvendt under forskellige scenarier for at teste, hvorledes programmet håndterer driftssituationer, hvor der muligvis forekommer skærpet opmærksomhed pga. individuelle krav til togsættene. I det følgende vil vi præsenterer to scenarier, hvor prototypen danner den samlede løsning for hægtning og parkering.

5.2.1 Første scenarie

Data for den første testkørsel stammer fra onsdag d. 26 februar 2014. Samlet set ankommer 29 togsæt til KH om aftenen, og 29 togsæt skal planlægges til den efterfølgende drift torsdag morgen. Vi

ved, at der ankommer 25 LHB og 4 LHF togsæt, og vi kender specifikationerne omkring antal tilbagelagte kilometer. Vi kan se, at 2 togsæt skal have påfyldt sand, mens der ikke forekommer situationer, hvor der er krav om vedligeholdelse. Efterspørgslen svarer til udbuddet af togtyper.

Det er relevant at nævne, at data allerede er hægtet og parkeret af planlæggeren, så vi ved, at der findes en mulig løsning. Vi har derfor et sammenligningsgrundlag[15].

Hægtning af togsæt til togn.				Tasks																										
				1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29
				Togn.	Togn.	Togn.	Togn.	Togn.	Togn.	Togn.	Togn.	Togn.	Togn.	Togn.	Togn.	Togn.	Togn.	Togn.	Togn.	Togn.	Togn.	Togn.	Togn.	Togn.	Togn.	Togn.	Togn.	Togn.	Togn.	Togn.	Togn.	Togn.
				72914	30214	60115	52215	52215	30115	22215	72915	72915	12215	30215	60216	52216	22216	30216	52117	52117	52217	32117	23118	32218	62218	62218	62118	32118	23219	13219	32219	23220
				Linie	Linie	Linie	Linie	Linie	Linie	Linie	Linie	Linie	Linie	Linie	Linie	Linie	Linie	Linie	Linie	Linie	Linie	Linie	Linie	Linie	Linie	Linie	Linie	Linie	Linie	Linie	Linie	Linie
				F2	C1	E1	H	H	C1	B1	F2	F2	A1	C1	E1	H	B1	C1	H	H	H	C2	B2	C2	E2	E2	E2	C2	B2	A2	C2	B2
				1	1	1	1	0.5	1	1	0.5	0.5	1	1	1	1	1	1	1	0.5	1	1	1	1	1	1	1	1	1	1	1	1
Obs	Nr.	Litra	Service																											
1	Togsæt 1	8201		LHB																										
2	Togsæt 2	8166			LHB																									
3	Togsæt 3	8193				LHB																								
4	Togsæt 4	8187					LHB																							
5	Togsæt 5	8192							LHB																					
6	Togsæt 6	4122						LHF																						
7	Togsæt 7	4112										LHF																		
8	Togsæt 8	8124								LHB																				
9	Togsæt 9	8110											LHB																	
10	Togsæt 10	8222													LHB															
11	Togsæt 11	4131																		LHF										
12	Togsæt 12	8198												LHB																
13	Togsæt 13	8168														LHB														
14	Togsæt 14	8146															LHB													
15	Togsæt 15	8191																LHB												
16	Togsæt 16	8183																	LHB											
17	Togsæt 17	8130																			LHB									
18	Togsæt 18	8144																				LHB								
19	Togsæt 19	8125																					LHB							
20	Togsæt 20	8666																						LHB						
21	Togsæt 21	8163																							LHB					
22	Togsæt 22	8127																								LHB				
23	Togsæt 23	8121																									LHB				
24	Togsæt 24	4110									LHF																			
25	Togsæt 25	8172																										LHB		
26	Togsæt 26	8202	S																										LHB	
27	Togsæt 27	8199																												LHB
28	Togsæt 28	8152																												
29	Togsæt 29	8190	S																											LHB

Figur 5.2 : Viser hægtningen mellem togsæt og tognumre

Vi ser på figur 5.2, at hægtningen tilnærmelsesvis placeres langs diagonalen, og derfor i tråd med LIFO-tilgangen. Programmet søger hermed en løsning, hvor sidst ankomne togsæt hægtes på først afgående tognummer. Tildelingen foregår under betingelse af, at efterspørgslen dækkes, hvorfor vi ser, at hægtningen af LHF afviger fra diagonalen. Samtidig ser vi, at togsæt 8102 (26) og 8190 (29) begge hægtes på servicelinjen, så de kan blive varetaget og få påfyldt sand (S=sand, V=vedligehold).

Vi har nu opstillet hægtningen, og programmet kan derfor foretage parkeringen af togsættene, som ses på figur 5.3. Her ser vi en mulig parkering, som overholder de betingelser, som er pålagt.

[15] Se i bilag 2

Parkering af togsæt på depot Vest							Depotspor						
							131	132	133	134	135	136	137
Dep.nr	Tog_ank	Ankomst	Tog_afg	Afgang	T	Service							
1	13657	19:03:00	13219	06:24:00	+	.	8199	.	.	.	.	.	.
2	32658	19:27:00	62218	06:12:00	+	.	8163	.	.	.	.	.	.
					++	.	8127	.	.	.	.	.	.
3	30502	00:48:00	52216	05:26:00	+	.	8168	.	.	.	.	.	.
4	72003	01:05:00	22215	05:10:00	+	.	8124	.	.	.	.	.	.
5	30603	01:17:00	72914	04:51:00	+	.	8201	.	.	.	.	.	.
13	52571	23:40:00	32117	05:59:00	+	.	.	8144	.	.	.	.	.
14	16502	00:52:00	52117	05:41:00	++	.	.	4131	.	.	.	.	.
	52501	00:20:00	52117	05:41:00	+	.	.	8183	.	.	.	.	.
15	60602	00:41:00	22216	05:30:00	+	.	.	8146	.	.	.	.	.
16	71057	19:15:00	72915	05:11:00	+	.	.	4110	.	.	.	.	.
	72003	01:05:00	72915	05:11:00	++	.	.	4112	.	.	.	.	.
24	23658	19:19:00	62118	06:15:00	+	.	.	.	8121	.	.	.	.
25	30602	00:57:00	12215	05:14:00	+	.	.	.	8110	.	.	.	.
28	13556	18:42:00	23220	06:40:00	+	S	.	.	.	8190	.	.	.
			32219	06:28:00	+	.	.	.	.	8152	.	.	.
29	32659	19:47:00	23118	06:07:00	+	.	.	.	.	8125	.	.	.
30	52500	00:00:00	52217	05:46:00	+	.	.	.	.	8130	.	.	.
31	16502	00:52:00	30215	05:18:00	+	.	.	.	.	8198	.	.	.
32	26603	01:09:00	52215	05:06:00	++	.	.	.	.	8187	.	.	.
	36503	01:08:00	52215	05:06:00	+	.	.	.	.	4122	.	.	.
37	62657	19:11:00	32118	06:19:00	+	.	.	.	.	.	8172	.	.
38	23659	19:39:00	32218	06:08:00	+	.	.	.	.	.	8666	.	.
39	52502	00:40:00	30216	05:38:00	+	.	.	.	.	.	8191	.	.
40	22502	00:56:00	60216	05:22:00	+	.	.	.	.	.	8222	.	.
44	32657	19:07:00	23219	06:20:00	+	S	.	.	.	.	.	8202	.
45	36503	01:08:00	30115	05:09:00	+	.	.	.	.	.	.	8192	.
48	16503	01:12:00	30214	04:58:00	+	.	.	.	.	.	.	.	8166
			60115	05:05:00	+	.	.	.	.	.	.	.	8193

Figur 5.3 : Viser hvornår togsættet ankommer, hvor det skal hægtes til, og hvor det skal parkeres.

Løsningen findes inden for kort tid, og parkering vil sikre afviklingen af morgendagens drift uden forstyrrelser. Afgangen mellem togsæt/togkompositioner på de enkelte depoter opfylder tidsrestriktionen på 10 minutter. Det fremgår at i tilfælde af, at to ankomne togsæt tildeles samme opgave, vil algoritmen prioriterer en samlet parkering fremfor en tidsforskydning mellem ankomsttiden. Dette ser vi ved parkeringen på depot 132, hvor ankomsttiden kun varierer med 1 minut.

5.2.2 Andet scenarie

Vi ser nu på en situation, hvor planlæggeren ønsker at sende udvalgte togsæt til vedligeholdelses- eller sandeftersyn, dvs. at der her har været et ønske om en genplanlægning til den første løsning. Prototypen har ved første kørsel fremstillet et løsningsforslag, som er blevet præsenteret for planlæggeren, se figur 5.3. Ved hjælp af beslutningsstøttesystemet kan han nu fiksere de togsæt, som han ønsker at omdirigere for derefter af køre programmet igen. Denne situation kan opstå hvis planlæggeren ser en fordel ved at sende ekstra togsæt til eftersyn i dag for at afværge flaskehalssituationer i de kommende dage. Han revurderer hermed løsningen. Muligheden for manuel omdirigering kan også benyttes i situationer, hvor eventuelle A,B eller C fejl er indmeldt. Figur 5.4 illustrerer resultatet på hægtningen:

Hægtning af togsæt til togur.				Tasks																											
				1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29
				Togur.	Togur.	Togur.	Togur.	Togur.	Togur.	Togur.	Togur.	Togur.	Togur.	Togur.	Togur.	Togur.	Togur.	Togur.	Togur.	Togur.	Togur.	Togur.	Togur.	Togur.	Togur.	Togur.	Togur.	Togur.	Togur.	Togur.	Togur.	Togur.
				72914	30214	60115	52215	52215	30115	22215	72915	72915	12215	30215	60216	52216	22216	30216	52117	52117	52217	32117	23118	32218	62218	62218	62118	32118	23219	13219	32219	23220
				Linie	Linie	Linie	Linie	Linie	Linie	Linie	Linie	Linie	Linie	Linie	Linie	Linie	Linie	Linie	Linie	Linie	Linie	Linie	Linie	Linie	Linie	Linie	Linie	Linie	Linie	Linie	Linie	Linie
				F2	C1	E1	H	H	C1	B1	F2	F2	A1	C1	E1	H	B1	C1	H	H	H	C2	B2	C2	E2	E2	E2	C2	B2	A2	C2	B2
				1	1	1	1	0.5	1	1	0.5	0.5	1	1	1	1	1	1	1	0.5	1	1	1	1	1	1	1	1	1	1	1	1
Obs	Nr.	Litra	Service																												
1	Togsæt 1	8201		LHB																											
2	Togsæt 2	8166	V & S							LHB																					
3	Togsæt 3	8193			LHB																										
4	Togsæt 4	8187				LHB																									
5	Togsæt 5	8192					LHB																								
6	Togsæt 6	4122						LHF																							
7	Togsæt 7	4112										LHF																			
8	Togsæt 8	8124							LHB																						
9	Togsæt 9	8110	S										LHB																		
10	Togsæt 10	8222									LHB																				
11	Togsæt 11	4131	V														LHF														
12	Togsæt 12	8198												LHB																	
13	Togsæt 13	8168														LHB															
14	Togsæt 14	8146													LHB																
15	Togsæt 15	8191																LHB													
16	Togsæt 16	8183	V																				LHB								
17	Togsæt 17	8130																	LHB												
18	Togsæt 18	8144																			LHB										
19	Togsæt 19	8125																				LHB									
20	Togsæt 20	8666																						LHB							
21	Togsæt 21	8163																							LHB						
22	Togsæt 22	8127																									LHB				
23	Togsæt 23	8121	V																										LHB		
24	Togsæt 24	4110																		LHF											
25	Togsæt 25	8172																										LHB			
26	Togsæt 26	8202	S																										LHB		
27	Togsæt 27	8199																						LHB							
28	Togsæt 28	8152																												LHB	
29	Togsæt 29	8190	S																												LHB

Figur 5.4 : Viser hægtningen mellem togsæt og tognumre

Den samlede løsning findes på under 2 minutter og der foretages en mulig parkering på baggrund af hægtningen. Som figur 5.4 illustrerer, vil parkeringen ændre sig i forhold til hægtningen ved første scenarie. Vi ser en løsning, der nu har taget højde for, at planlæggeren ønskede yderligere 5 omdirigeringer af togsæt til servicelinjen.

Parkering af togsæt på depot Vest							Depotspor						
							131	132	133	134	135	136	137
Dep.nr	Tog_ank	Ankomst	Tog_afg	Afgang	T	Service							
1	32657	19:07:00	13219	06:24:00	+	S	8202	.	.	.	.	.	.
2	23659	19:39:00	62218	06:12:00	++	.	8666	.	.	.	.	.	.
	32658	19:27:00	62218	06:12:00	+	.	8163	.	.	.	.	.	.
3	32659	19:47:00	32117	05:59:00	+	.	8125	.	.	.	.	.	.
4	52571	23:40:00	52217	05:46:00	+	.	8144	.	.	.	.	.	.
5	60602	00:41:00	60216	05:22:00	+	.	8146	.	.	.	.	.	.
13	13657	19:03:00	32218	06:08:00	+	.	.	8199	.	.	.	.	.
14	52500	00:00:00	52117	05:41:00	++	.	.	8130	.	.	.	.	.
	71057	19:15:00	52117	05:41:00	+	.	.	4110	.	.	.	.	.
15	16502	00:52:00	22216	05:30:00	+	V	.	4131	.	.	.	.	.
			30215	05:18:00	+	.	.	8198	.	.	.	.	.
16	36503	01:08:00	52215	05:06:00	+	.	.	4122	.	.	.	.	.
					++	.	.	8192	.	.	.	.	.
24	52501	00:20:00	23118	06:07:00	+	V	.	.	8183	.	.	.	.
25	72003	01:05:00	30115	05:09:00	+	.	.	.	8124	.	.	.	.
28	32658	19:27:00	62118	06:15:00	+	.	.	.	.	8127	.	.	.
29	30502	00:48:00	52216	05:26:00	+	.	.	.	.	8168	.	.	.
30	22502	00:56:00	72915	05:11:00	+	.	.	.	.	8222	.	.	.
	72003	01:05:00	72915	05:11:00	++	.	.	.	.	4112	.	.	.
31	30603	01:17:00	72914	04:51:00	+	.	.	.	.	8201	.	.	.
37	13556	18:42:00	23220	06:40:00	+	S	.	.	.	.	8190	.	.
			32219	06:28:00	+	.	.	.	.	.	8152	.	.
38	16503	01:12:00	22215	05:10:00	+	V & S	.	.	.	.	8166	.	.
			30214	04:58:00	+	.	.	.	.	.	8193	.	.
44	23658	19:19:00	23219	06:20:00	+	V	.	.	.	.	.	8121	.
45	26603	01:09:00	60115	05:05:00	+	.	.	.	.	.	.	8187	.
48	62657	19:11:00	32118	06:19:00	+	.	.	.	.	.	.	.	8172
49	52502	00:40:00	30216	05:38:00	+	.	.	.	.	.	.	.	8191
50	30602	00:57:00	12215	05:14:00	+	S	.	.	.	.	.	.	8110

Figur 5.5 : Viser hvornår togsættet ankommer, hvor det skal hægtes til, og hvor det skal parkeres.

Vi ser her, hvorledes operatøren manuelt kan fiksere togsæt til service, såfremt han mener, at det vil gavne planlægningen i det lange løb. Vi ser, at depoterne bliver fyldt op, så det sikres, at der er en tidsforskydning på minimum 10 minutter mellem parkeringen af ankomne togsæt og afgående tognumre.

Tilsvarende testkørsler er foretaget, og det generelle billede er, at en mulig parkering, som opfylder de nedsatte kriterier, ofte findes. Løsningseffektiviteten viser sig hermed at være høj, og planlæggeren vil i prototypens resultat kunne vurdere opsætningen og eventuelt genplanlægge.

5.2 Sammenligning med BSS

For at sammenligne vores generede løsningsforslag med den manuelt udførte operatøropgave skal vi have for øje, hvad det er, vi samlet set søger. Med udgangspunkt i hægtningen på den enkelte dag er målsætningen primært at dække efterspørgslen på alle togløb og sikre, at alle togsæt ikke overskrider deres individuelle restriktioner. Vi så at det originale scenarie dækkede over to enheder, der er ønsket sandpåfyldt, og at ingen enheder skal til vedligeholdelsessyn. Problemet i hægtningen er balanceret mellem udbud og efterspørgsel og danner således ingen problematikker for hverken MAS eller vores model. Parkeringen ser også tilforladelig ud for planlæggeren i MAS, men der iagttages, at det påkrævede interval på 10 minutter ikke er overholdt imellem individuelle togsæt. Dette er løst indenfor en relativ kort tid i den modelgenerede løsning, hvilket, vi mener, er en mere robust parkering end planlæggerens.

Omvendt opstår der en ulempe, som sker ved generalisering af planlæggerens rolle. I modellen ønsker vi at se udgående togkompositioner som samlede deponeringer. Her ser vi, at vores model tillader en parkering mellem to togsæt, der ankommer med et minuts interval, fordi de skal afgå sammen som en togkomposition. Dette kan være problematisk at nå for depotbemandingen.

Alt i alt kommer vi tæt på planlæggerens originale løsning inden for en yderst efficient beregningstid i forhold til de nuværende forhold. Kun med små ændringer i den fundne parkering vil vi derfor både have en mere robust parkering, et skærpet fokus på rettidig omdirigering og et besparelseselement i planlægningsprocessen.

6. Perspektivering

6.1 Diskussion

Formålet med hægtningsproblemet er at finde en mulig tildeling af ankomne togsæt til afgående togløb. Det er her essentielt, at de togsæt, som kræver eftersyn i form af vedligeholdelse, sandpåfyldning eller med fejl, bliver omdirigeret til servicelinjen. Disse forbehold betyder, at hægtningsproblemet ikke er trivielt, men at det oftest vil være muligt at finde en samlet løsning. I hægtningsproblemet tildeles de ankomne togsæt efter LIFO-teorien, hvor der tages højde for ankomsttid i forhold til afgangstid den efterfølgende dag, dog stadig med forbehold for eventuelle eftersyn. Afhængigt af hægtningen og antallet af omdirigerede togsæt, der har behov for service, øges problematikken omkring parkeringsproblemet. Formålet er her at parkere de ankomne togsæt, som gennem hægtningsproblemet er blevet tildelt til et specifikt afgående tognummer. Her øges kompleksiteten, da parkeringen skal sikre, at der ikke forekommer krydsninger, hvor togsæt blokerer for hinanden, da afgangstiden for forreste togsæt er senere end det bagved holdende. Ydermere skal kapaciteten på de enkelte depotspor overholdes, og det optimale scenarie i følge DSB S-tog vil være at tilgodese, at ankomne togkompositioner ikke splittes op og parkeres på forskellige depotspor.

Vores opdeling i en to-trins løsning, hvor shunting processen løses separat, er inspireret af en række litterære artikler heriblandt Freling et al. [FLKH02], Lentink et al. [LFKW03] og Stefano et al. [SK03]. Gennem litterturen har vi også set forsøg på globale løsningsalgoritmer, bl.a. fra Schrijver et al. [SLK05]. Her er løsningsresultatet konstateret dårligt og må forkastes. [SLK05] henviser specielt til uacceptable beregningstider. Dette er ikke enestående i litteraturen omkring shunting problemet og bliver ligeledes påpeget i [WZ00], hvor de fremviser fire forskellige løsninger på det samlede problem for sporvogne.

Den to-trins løsning, som vi opstiller i prototypen, tager i modsætning til [FLKH02] og [LFKW03], der forsøger at minimere opsplit af togkompositioner, udgangspunkt i at optimere en interaktion imellem de to trin, således at vi sikrer en parkering af flest mulige togsæt. Den sekventielle søgning for en løsning medfører en risiko for, at vi ikke finder en løsning på trods af dens tilstedeværelse. Derfor har vores målsætning været at skabe en løsningsorienteret søgealgoritme for efterfølgende at tage højde for at minimere opsplit af togkompositioner. Der kan være flere årsager til den skærpede fokus på opsplit, som bl.a. forekommer i den hollandske litteratur, og vi vil mene, at det i høj grad hænger sammen med et øget antal af togkompositioner, som det hollandske tognetværk skal

varetage i deres parkeringsproblem. Vi ser også i artiklen [SLK05], at de forsøger at minimere antallet af forskellige togtyper på samme spor, hvilket kunne antyde, at deres grundlag for parkeringen er mere komplekst end for DSB S-tog.

Vi har opbygget en prototype, der tilgodeser en hægtning, som sikrer det bedste udgangspunkt for en parkering samt sikrer en glidende drift. Grundlaget for dette skabes ud fra den eksponentielt stigende omkostning, som er forbundet med selve hægtningen, hvor tildelingen søger diagonalen i LIFO-matricen.

Som et nyt element tillader vi, at DSB S-tog i tilfælde af manglende materiel af en specifik type kan gennemføre driften med over- eller underkapacitet. Dette er en dyr omkostning og ikke et scenarie, der foretrækkes, men det sikrer, at operatøren kan sende et LHF togsæt til eftersyn, selvom servicelinjen efterspørger kapaciteten fra et LHB togsæt. Dette er et aspekt, som DSB har haft et ønske om at få inddraget i modellen. Vi ser bl.a. i Dirksen [D10], at denne håndtering ikke er implementeret, hvilket resulterer i, at man må antage, at antallet af efterspurgte typeenheder er lig den udbudte mængde.

Med baggrund i artiklen af Wezel et al. [WJ08] har vi forsøgt at belyse, hvorledes vi mener, at den bedste metode til at oprette en beslutningsstøttesystem (BSS) foregår ved en koalition imellem computer og operatør. Det fremgår i artiklen, at det gælder om at udnytte hver komponents ressourcer bedst muligt. Det er en tilgang vi er enige i, hvorfor vi har forsøgt at implementere denne tankegang i prototypen.

Den opgave, der ligger i at varetage TUSP hos DSB S-tog, inkluderer bl.a. ansvaret for at sende togsæt rettidigt til vedligeholdelseseftersyn samt sandpåfyldning. I den sammenhæng mener vi, at ansvaret i første omgang skal varetages af den maskinelle supportkraft, da den har de bedste forudsætninger for at varetage det samlede problemfelt. Vi mener, at den menneskelige operatør har en relativt stor sandsynlighed for at overse delelementer i et så stort og kompleks scenarie, hvorimod man bør udnytte operatørens viden, intuition og erfaringer andre steder i processen for samlet set at effektivisere opgaven.

Der vil opstå situationer, hvor computersupporten ikke kan levere en fuldstændig løsning. Her vil det være op til operatøren manuelt at håndtere problematikken og finde en løsning. Som [WJ08] skriver, bør man netop inddrage operatøren i tilfælde af en ufuldstændig løsning, hvor han manuelt kan

påvirke parameterværdierne i modellen og derved sikre en løsning til problemet. Det er netop her, at samspillet mellem operatørens kognition og viden samt den maskinelle information skal bidrage til en løsning.

I de tilfælde hvor prototypen ikke er i stand til at finde en løsning, kan årsagen være, at vi ikke kan sikre, at der forestår et 10 minutters tidsinterval imellem hvert togsæt på depotsporene. Ved analyser af faktiske depotplaner har vi observeret, at denne parameterværdi ikke nødvendigvis overholdes, men derimod relaxeres i praksis hos DSB S-tog[16]. Det betyder, at operatøren har mulighed for at sænke tidsintervallet i modellen for at opnå en løsning eller manuelt indflette de manglende parkeringer på depoterne.

Et andet element, hvor vi også ønsker at inddrage operatøren, er genplanlægning af allerede løste scenarier. Denne form for alternative løsningsforslag til genplanlægning beskrives også i [WJ08]. Her mener vi netop, at den menneskelige intuition ikke kan indfanges i en model, og at operatøren og hans viden skal være et led i at sikre, at der ikke opstår flaskehalssituationer på sigt på værkstederne. Prototypen har implementeret denne del. Her vil det være muligt, at genplanlægge den oprindelige løsning, hvor operatøren nu kan omdirigere togsæt, som ligger i nærheden af kravet om serviceeftersyn. Togsæt som ikke er indfanget af modellens restriktioner, men som operatøren ser fordele ved at omdirigere.

6.2 Videre arbejde

Med hensyn til prototypen, de algoritmer og metoder der er beskrevet i denne afhandling, findes der flere retninger, som er værd at udforske yderligere. En central del af opgaven har været at identificere og definere rammerne omkring problemet. Dette er et vanskeligt arbejde i sig selv, og der er lang vej til en komplet løsning. Vi vil i dette afsnit fremstille andre vinkler til vores model, som vil kunne øge effektiviteten og skabe et bedre grundlag, såfremt modellen udvides til at indeholde flere specifikke restriktioner for de enkelte togsæt.

[16] Se bilag 2 for originalt scenarie. Her forekommer tidsintervaller ned til 3 min mellem to parkeringer.

6.2.1 Opsplit

En udvidelse af modellen kan samtidig betyde en forøget beregningstid. Som en del af to-trins løsningen ville det være interessant at implementere et element, der kunne tildele objektfunktionen en bonus, såfremt den parkerede togkompositioner på samme depot, men ikke som et krav. Problematikken opstår på nuværende tidspunkt, da vi ønsker, at algoritmen samtidig skal tage højde for det fastlagte tidsinterval, som netop ikke skal være gældende overfor ankomne togkompositioner.

6.2.2 Maksimering af tidsinterval

Vi har tidligere refereret til et ønske om en robust køreplan, som kan modstå mindre forstyrrelser i driften. For parkeringen betyder det et implicit ønske om at maksimere tidsintervallet mellem afgange og ankomster. Herved vil små forskydninger i køreplanen ikke påvirke udrulningen af togsæt eller den planlagte parkering ved ankomne togsæt.

Dette kan håndteres, såfremt vi ved, at der er en komplet løsning til problemet. Vi ved hermed, at det er muligt at parkere alle togsæt ud fra et givent tidsinterval. Det vil derfor være muligt at maksimere det mindste tidsinterval, der forekommer mellem alle parkeringer. Fordelen er, at jo større interval vi kan opnå mellem parkeringerne, desto mere robust vil planlægningen blive.

6.2.3 Inddragelse af weekenddage

Prototypen tager udgangspunkt i planlægningen af shunting problemet i hverdagene. Denne begrænsning er sat, for at kunne fastholde delelementer i et samlet BSS, som skal være brugbart og afspejle de enkelte opgaver. Dette medfører visse mangler, og i dette tilfælde en løsning, som tager højde for alle ugens dage. Vi har udviklet et system, der fastlægger processerne omkring de travle hverdage, hvorfor de resterende bør være langt nemmere at tilgå.

Prototypen tager på nuværende tidspunkt højde for planlægningen af driften tirsdag-fredag. Dvs. at MAS i løbet af en mandag skal foretage hægtningen og parkeringen til tirsdagens drift. Planlægningen af driften fra lørdag-mandag mangler hermed at blive implementeret.

En implementering af planlægningen af driften lørdag samt parkering fredag nat vil ikke kræve større tilføjelser i programmet. Problematikken opstår i, at der lørdag dag sættes et reduceret antal tognumre i drift, da efterspørgslen er væsentligt reduceret. Det betyder, at der er et specifikt antal togsæt, der ikke skal sættes i drift. Disse skal derfor placeres bagerst på depotsporet under parkeringen fredag, så de ikke forstyrrer udrulningen lørdag morgen. En tilgang til løsningen af dette setup kan være at tilpasse det datainput, som informerer omkring ankomst og afgang. Denne tilpasning indbefatter en tilføjelse af en såkaldt dummy-variabel i form af et fiktivt tognummer, som skal "varetages" af de togsæt, der ikke skal benyttes i næste dags drift. Det er vigtig, at afgangstiden ligger senere end de reelle tognumre, samt at vi sikrer en balance mellem udbud og efterspørgsel af ankomne togsæt og afgående tognumre.

Karakteristika for søndagens og mandagens drift skal ligeledes anskues som to individuelle scenarier. Antallet af ankomne togsæt og afgående tognumre vil variere, og der skal derfor opstilles forskellig datamanipulation til de enkelte dage, så vi ved hjælp af dummy-variable kan balancere udbuddet med efterspørgslen.

6.2.4 Relaxering af kilometerrestriktion

Alle virksomheder er indbefattet af forretningsregler, og DSB S-tog er ingen undtagelse. En af disse forretningsregler munder ud i vedligeholdelseseftersyn, som vi gentagne gange har omtalt, hvor det er dikteret fra Transportministeriet, at et togsæt ikke må overstige 50 Mm inden vedligeholdelse. Dette krav resulterer i, at vores model ikke sender disse enheder i drift, og hvis der ikke er tilstrækkelige togsæt, ender vi ud i en uløselig situation i hægtningsprocessen. Det strenge krav er principielt 55 Mm, men de resterende 5 Mm er ikke genstand for optimering. Praktisk set betyder det, at hvis togsættet kommer i dette interval, skal det synliggøres, så MAS kan sikre omgående service. Men det betyder samtidig, at de tidligere antagelser om nultolerance ved 50 Mm kan lempes.

Dette kan håndteres ved at foretage en tilføjelse under vedligeholdelsesrestriktionen i modellen. Foruden begrænsningen, hvor vi tidligere kun så på 50 Mm, skal vi nu skabe en mulighed for at tillægge togsættene yderligere 5 Mm. Dette kan gøres ved at inddrage et ekstra sæt af beslutningsvariable, som tillader en overskridelse af kilometergrænsen, dog på bekostning af en meromkostning. Restriktionen bliver således:

$$\alpha_j + \sum_{i \in I} x_{ij}^m * \delta_i \leq 50Mm + u_j * 5Mm \qquad \forall\, j \in J$$

Hvor u_j er binær og beskriver, hvorledes togsæt *j* er i den kritiske zone. Hvis den kritiske zone opnås, er det således op til MAS enten at fiksere togsættet over på servicelinjen ved dagens start eller på anden vis sikre, at service indtræder, inden de resterende kilometer er opbrugt.

Vi er nødt til at inddrage u_j i objektfunktionen, så der tages højde for meromkostningen.

6.2.5 Fiksering generelt/Fiksering af andre linjer

En skærpet kontrol over specifikke togsæt drejer sig ikke nødvendigvis kun om serviceeftersyn. DSB S-tog har også et ønske om at kunne sende bestemte togsæt ud i konkrete togløb, da de nu og da kører forskellige reklamekampagner, eller hvis de ønsker en passageroptælling på specifikke strækninger. Det skal derfor være muligt at fiksere bestemte togsæt rundt på konkrete strækninger i netværket. Dette kan løses ved en integration med det nuværende program. Vi har allerede belyst, hvorledes prototypen kan håndtere manuelle omdirigeringer til servicelinjerne, og tankegangen er den samme her. Der skal nu opbygges en brugerflade, som tillader operatøren at træffe en række beslutninger inden programmet afvikles. Det kan bl.a. være beslutninger omkring hvilke togsæt, der ønskes omdirigeret til bestemte linjer. Prototypen skal herefter kunne håndtere omdirigeringen for de enkelte togsæt, som blot skal fikseres til en hægtning af det tilhørende tognummer, *i*, som er tilknyttet den ønskede strækning. Denne restriktion kunne have følgende form:

$$\sum_{\substack{i \in I \\ \text{ønskede} \\ \text{strækning}}} x_{ij}^m * h_j \leq t_j \quad hvor\ h_j = \begin{cases} 1\ hvis\ manuelt\ fikseret \\ 0\ ellers \end{cases} \qquad \forall\, j \in J$$

t_j er en binær beslutningsvariabel, der optræder i objektet, hvormed der kan tillægges en bonus ved en given fiksering af togsæt *j*.

6.3 Økonomiske besparelser

Dag-til-dag planlægningen foregår på nuværende tidspunkt manuelt hos DSB S-tog. Vi har igennem opgaven påpeget, at vi ikke finder det optimalt, da planlæggeren bag den manuelle proces kan miste det samlede overblik. Ved ikke at automatisere mulige delprocesser kan resultatet føre til en række

ekstra driftsomkostninger. I afsnit 1.4 omtalte vi, hvilke økonomiske aspekter, vi mener, et beslutningsstøttesystem kan bidrage til at nedbringe.

6.3.1 Besparelse ved minimering af eftersyn

Vi mener, at DSB S-tog kan finde en økonomisk gevinst ved at kombinere ekspertviden og maskinel software, hvorved de bibeholder det samlede overblik i netværket. Formålet med at lave et BSS til dag-til-dag planlægningen hos MAS er ifølge DSB S-tog et led i et projekt omkring rettidig omdirigering af togsæt til eftersyn. Prototypen sikrer ikke en løsning, hvor togsættene som førsteprioritet kører så langt som muligt i forhold til kilometergrænsen. Dette kunne have været en målsætning, men det ville samtidig være på bekostning af køreplanens robusthed. Programmet sikrer derimod, at togsættene tildeles service rettidigt. Den økonomiske gevinst indfries i takt med den styring, der ligger bag BSS og kombinationen af, at den kognitive planlægger frit kan omdirigere togsæt til serviceeftersyn, såfremt han ser en fordel, som effektiviserer den fremtidige drift. Hermed bevares det brede perspektiv, og DSB S-tog undgår vedligeholdelseseftersyn af togsæt, der endnu ikke har kørt op til den tilladte grænse. Denne problematik kommer til udtryk i figur 6.1, hvor DSB S-tog har erfaret, at der løbende forekommer alt for tidlige omdirigeringer af togsæt, da de ansvarlige ønsker at begrænse tilfælde af togsæt, der overskrider kilometerrestriktionen.

Alle DSB S-togs vedligeholdelseseftersyn har omkostninger, hvilket bl.a. kommer til udtryk i de direkte omkostninger forbundet med selve eftersynet. I tabel 6.1 nedenfor har vi liste nogle af de større og omkostningstunge eftersyn. Beløbene er indhentet fra DSB S-tog og er approximerede udgifter:

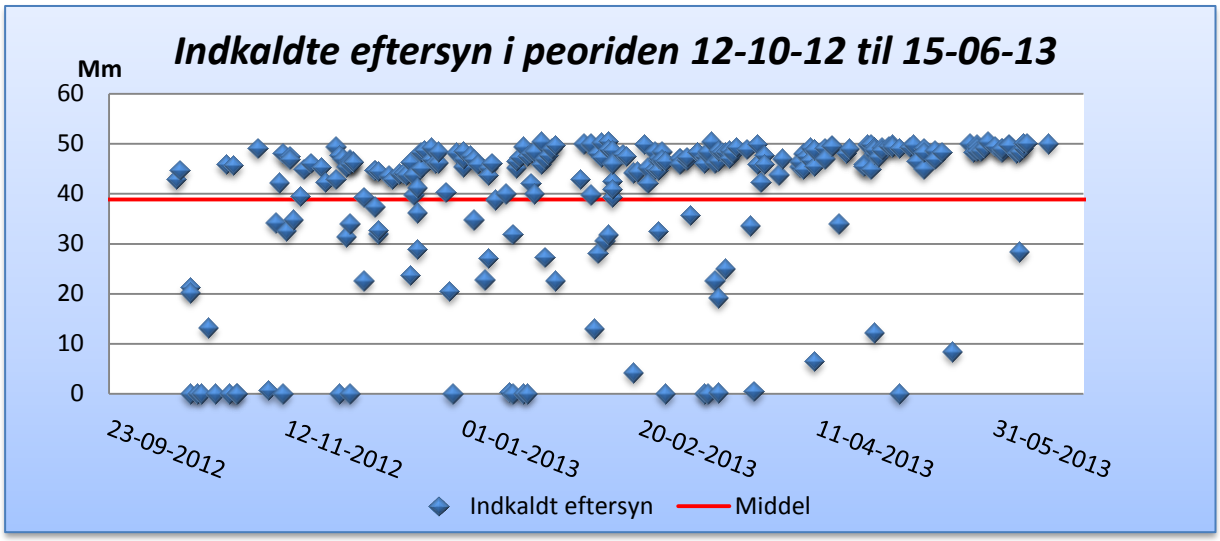
SA/SE	Pris
SA 1200 Mm eftersyn	1.709.000
SA 1800 Mm eftersyn	1.338.000
SA 2400 Mm eftersyn	1.590.000
SE 800 Mm eftersyn	677.000

Tabel 6.1 : Approx. prisoversigt for 4 typer af vedligeholdelsessyn

Når vi omtaler et enkelt 50 Mm vedligeholdelseseftersyn, så vil omkostningerne forbundet hertil ikke være en del af oversigten i tabel 6.1. Dette skyldes, at et normalt eftersyn ikke er en stor økonomisk udgift på kort sigt. Vigtigheden for dette element i prototypen er ikke desto mindre essentielt. Når et togsæt sendes til eftersyn, hvad end det har tilbagelagt 40 Mm, 42 Mm eller 51 Mm, så vil systemet

registrere en samlet kilometertæller på 50 Mm for det pågældende togsæt. På langt sigt vil det derfor have en større betydning, såfremt DSB S-tog ”mister” kilometer på hver af deres 135 togsæt ved samtlige eftersyn. Det betyder, at DSB S-tog samlet set vil kunne reducere antallet af store og omkostningsfulde eftersyn ved at sætte et fokus på, at de togsæt, der sendes til eftersyn, ligger tilstrækkelig tæt på kilometergrænsen.

Figur 1.6.1 illustrerer data over togsæt indkaldt til vedligeholdelseseftersyn i perioden 12.10.2012 til 15.06.2013. Vi ser her en klar tendens, som indikerer, at eftersynene forekommer for tidligt i forløbet.

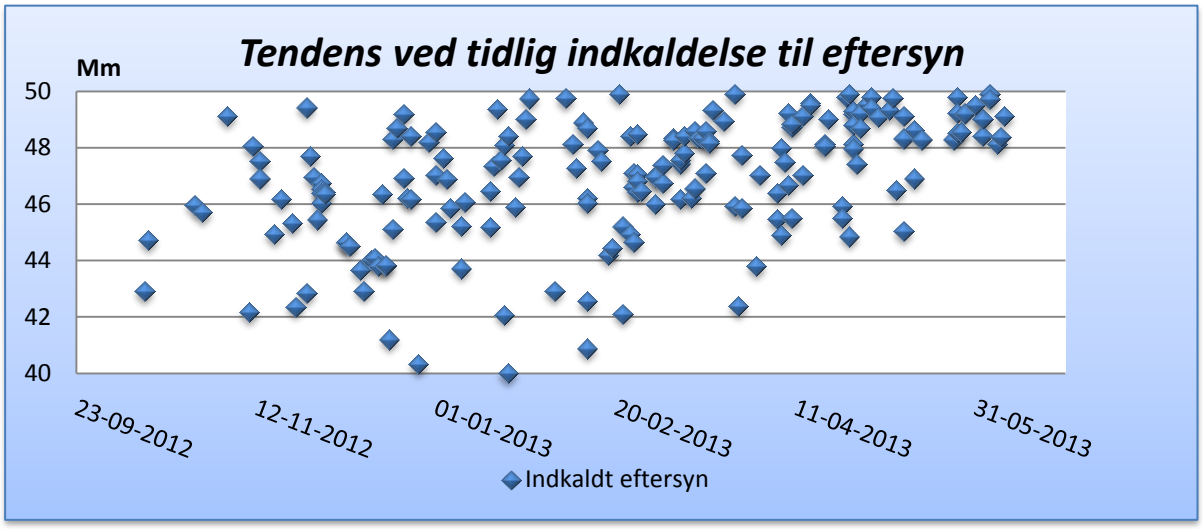


Figur 6.1 : Viser antal indkaldelser, der fandt sted i tidsrummet 12-10-2012 til 15-06-2013.

Data er leveret fra DSB S-tog, og de har bekræftet, at observationer liggende omkring 0 Mm skal antages som fejldata. Vi har valgt ikke at fjerne disse outliers fra oversigten for ikke at komme til at manipulere unødvendigt med billedet.

Vi ser, at der forekommer mange eftersyn imellem 20 Mm og 50 Mm i slutningen af 2012 og starten af 2013, og årsagen kan bl.a. forklares ved, at der før i tiden var en tidsrestriktion sideløbende med kilometerrestriktionen. Ifølge DSB S-tog er den udvikling, tabel 6.1 viser et resultat af et stigende fokus på at effektivisere indkaldelsesprocesserne, så grænsen ikke blev overskredet. På trods af at vi ikke kan forklare årsagen til de bemærkelsesværdige afvigelser, kan vi konkludere, at de enkelte eftersyn ikke samler sig tæt nok omkring den tilladte grænseværdi.

Grafen i tabel 6.2 synliggør periodens indkaldelser, som er foretaget mellem 40 Mm og 50 Mm. Her ses det tydeligt, at der forekommer en uacceptabel tendens, hvor togsæt sendes for tidligt til eftersyn.



Figur 6.2 : Oversigt over indkaldte eftersyn af togsæt der har tilbagelagt mellem 40 Mm – 50 Mm.

Det er stort set umuligt for os, at opsætte et konkret besparelsesbeløb, som DSB S-tog kan vinde ved en implementering af et BSS. Men det er tydeligt, at der kan optimeres på den proces, der ligger ved rettidig eftersyn. Et BSS vil kunne medvirke til at styrke den planlægning, operatøren står overfor. Ydermere kan et BSS være med til at styre processen omkring togsættenes tilbagelagte kilometerdistance, så det på lang sigt vil være muligt, at minimere de omkostninger, der er forbundet med regelmæssige eftersyn.

6.3.2 Operationelle besparelse i planlægningsprocessen.

Et naturligt besparelsespotentiale ved at overgå fra manuelle til fuld- eller semiautomatiske processer er umiddelbart bemandingen af de nuværende arbejdsopgaver. Herved vil enhver form for brugbart hjælpeværktøj have en besparende effekt. Denne effekt vil kunne måles, såfremt det ikke længere er nødvendigt for en planlægger at sidde og foretage den fysiske hægtning og parkering. Dette vil vi ikke yderligere kommentere, da det altid vil være et resultat af automatisering.

Vi synes derimod, at det er interessant for DSB S-tog at se på depotplanlægning i den virtuelle planlægningsfase. Ved brug af prototypen kan denne proces blive overflødig. Dette skyldes, at vi nu

varetager både parkering af togsæt til tognumre under parkeringsproblemet simultant i dag-til-dag planlægningen. Vi benytter derfor ikke længere den depotplanlægning, der foreligger, hvor det på forhånd er fastlagt hvilket tognumre, der besætter bestemte depotpladser.

7. Konklusion

Formålet med dette projekt var, at udvikle en prototype, der sikrer en effektiv planlægning af hægtning mellem togsæt og togløb, samt parkeringen over natten på Københavns Hovedbanegård. Problemet er tæt knyttet til det klassiske Train Unit Shunting Problem (TUSP), som er et omdiskuteret emne i den litterære verden.

For at kunne opnå viden omkring TUSP, generel togplanlægning og beslutningsstøttesystemer har vi studeret flere litterære artikler publiceret forskellige steder i Europa. I kombination med viden omkring DSB S-tog har dette været grundlaget for udviklingen af vores prototype. DSB S-tog har igennem de seneste år haft øget fokus på at optimere forskellige elementer i deres planlægningsproces, hvor de blandt andet ønsker en forbedring af deres dag-til-dag planlægning uden forstyrrelser, hvor de har erfaret, at visse delprocesser ikke forløber optimalt, heriblandt omdirigeringen af togsæt til vedligeholdelse.

Løsningen til problemet er opnået ved at udvikle et beslutningsstøttesystem, der kan ses som et automatisk værktøj, der vejleder planlæggeren. Det har vist sig, at det er muligt at danne en to-trins løsning vha. SAS, der hægter og parkerer de ankomne togsæt. Prototypen indeholder flere parametre, som kan justeres under optimeringsproceduren, hvilket øger kvaliteten. Vi har gennem tests og udvikling fundet beregningstiden yderst efficient, hvor løsningen oftest præsenteres efter få minutter. I modsætning til fx [FLKH02] har vi forsøgt at bibeholde den polynomiale beregningstid ved at bevare lineariteten i problemstillingen i parkeringsproblemet. Dette resulterer i, at der er en del elementer, som operatøren manuelt skal forholde sig til, og som derved ikke er genstand for optimering. I tråd hermed, har vi ikke arbejdet videre med den globale løsningsalgoritme, eftersom tidskompleksiteten tilsvarende er NP-komplet.

Man vil altid kunne finde fordele og ulemper ved at automatisere og generalisere processer, men det er vores erfaringer, at koalitionen mellem computer og bruger skaber de bedste resultater. På trods af, at vi ikke opnår en global optimal løsning, så formår vores BSS at præsentere operatøren for en løsning, som han efterfølgende kan vurdere og optimere yderligere. Her forekommer en effektivisering af køreplanlægningen, hvor de beslutninger, der tages, ikke blot sikrer en glidende drift dag-til-dag, men hvor kombinationen af BSS og operatør skaber en mere robust plan set over en længere periode.

DSB S-tog har et ønske om at minimere deres driftsomkostninger, og det er vores klare overbevisning, at et BSS vil kunne bidrage hertil. Vi har vist, at prototypen er i stand til at fremlægge en plan, der tager højde for de informationer, som er tilgængelige omkring de enkelte togsæt, hvormed operatøren informeres i tilfælde af et givent krav omkring vedligeholdelse. Den øgede fokus og den mulighed, planlæggeren har for at omdirigere enhederne, skal sikre, at togsæt i fremtiden vil blive sendt på værksted rettidigt. Hermed vil det på lang sigt være muligt at minimere antallet af større eftersyn og dermed sænke omkostningerne, som er forbundet med vedligeholdelse.

8. Litteraturliste

[BAM08] Rainer E. Burkard, Mauro Dell'Amico, Silvano Martello. Assignment Problems, 2008

[E46] Thomas E. Easterfield, A combinatorial algorithm, *J. London Math. Soc.,* 21 Jan 1946 (p:219-226).

[F06] Gunnar Frost, Noter ril Kombinatorik og grafteori, Københavns Universitet, Matematisk Afdeling, februar 2006

[D10] Jakob Dirksen, Speciale Course, Parking and Litra Assignment for S-tog, DTU, July 2010

[DSB13] DSB Årsrapport 2013

[FLKH02] Richard Freling, Ramon M. Lentink, Leo G. Kroon, and Dennis Huisman. Shunting of Passenger Train Units in a Railway Station. Technical Report ERS-2002-074-LIS, Erasmus University Rotterdam, Rotterdam, 2002.

[G08] Julie Jespersen Groth, Decision Support for the Rolling Stock Dispatch, DTU, PhD 2008

[HKLV05] Dennis Huisman, Leo G. Kroon, Ramon M. Lentink, and Michiel J.C.M. Vromans. Operations Research in Passenger Railway,Transportation. Technical Report ERS-2005-023-LIS, Erasmus University Rotterdam, Rotterdam, 2005.

[K02] Keiding, Hans; Operations Analyse, 1st ed. 2002, (p. 223)

[K55] H.W. Kuhn, The hungarian method for the assignment problem, *Naval Res. Log. Quart.,* 2:83-97, 1955.

[KHSB02] Thomas Kvist, Peter Hellstr¨om, Bengt Sandblad, and Jens Bystr¨om. Decision Support in the Train Dispatching Process. 2002.

[KLS06] Leo G. Kroon, Ramon M. Lentink, and Alexander Schrijver, Shunting of Passenger Train Units: and Integrated Approach, Transportation. Technical Report ERS-2005-023-LIS, Erasmus
University Rotterdam, Rotterdam, 2005.

[LEN06] Ramon M. Lentink. Algorithmic Decision Support for Shunt Planning. PhD thesis, Erasmus University Rotterdam, Rotterdam, February 2006.

[LFKW03] Ramon M. Lentink, Pieter-Jan Fioole, Leo G. Kroon, and Cor van ’t Woudt. Applying Operations Research techniques to planning of train shunting.

Technical Report ERS-2003-094-LIS, Erasmus University Rotterdam, Rotterdam, 2003.

[R02] Colin Robson, Real world research, 2nd ed. 2002, (p. 178)

[S92] Stephen F. Smith, Knowledge-Based Production Management: Approaches, Result and Prospects SAS Institute, Center for integrated Manufacturing Decision Systems The Robotics Institute Carnegie Mellon University, 1992

[SAS11] SAS Institute Inc. 2011. SAS/OR® 9.3 User's Guide, Mathematical Programming. Cary, NC: SAS Institute Inc., 2011. (P: 227-274).

[SK04] Gabriele Di Stefano and Magnus Love Koˇci. A Graph Theoretical Approach To The Shunting Problem. Electronic Notes in Theoretical Computer Science , 92:16–33, 2004.

[SLK05] Alexander Schrijver, Ramon M. Lentink, and Leo G. Kroon. Shunting of Passenger Train Units: an Integrated Approach. Technical report, Erasmus University Rotterdam, Rotterdam, 2005.

[WB02] Wout van Wezel, Bas Barten, Cognition, tasks and planning: supporting the planning of shunting operations at the Netherlands Railways, Faculty of Economics and Business, University of Groningen,, The Netherlands, 2008.

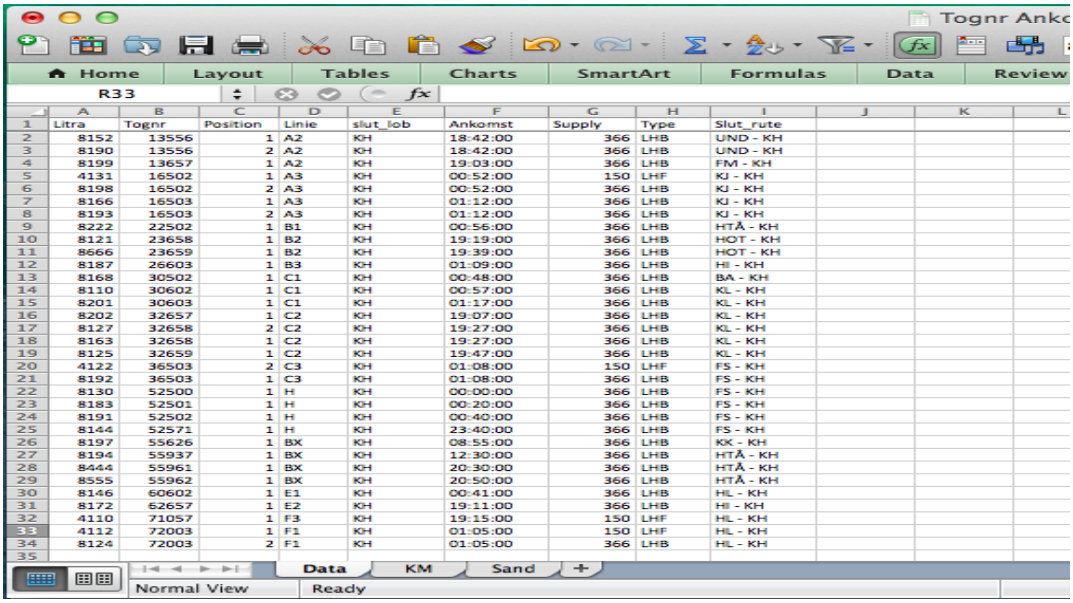
[WB02] Wout van Wezel, Rene, Hierarchical Mixed Initiative Planning Support, Faculty of Management and Organization, University of Groningen, The Netherlands, 2002.

[WZ00] Thomas Winter and Uwe T. Zimmermann. Real-time dispatch of trams in storage yards. Annals of Operations Research , 96:287–315, 2000.

Bilag

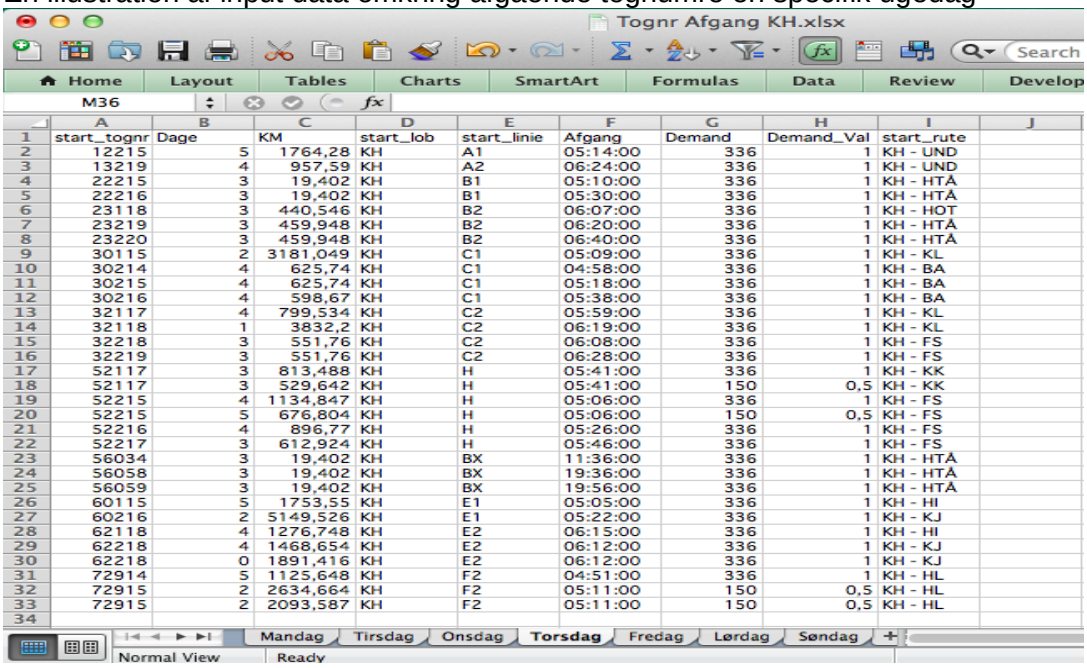
Bilag 1 - Data over ankomst og afgang

En illustration af input data omkring de ankomne togsæt:



Litra	Tognr	Position	Linie	slut_lob	Ankomst	Supply	Type	Slut_rute
8152	13556	1	A2	KH	18:42:00	366	LHB	UND - KH
8190	13556	2	A2	KH	18:42:00	366	LHB	UND - KH
8199	13657	1	A2	KH	19:03:00	366	LHB	FM - KH
4131	16502	1	A3	KH	00:52:00	150	LHF	KJ - KH
8198	16502	2	A3	KH	00:52:00	366	LHB	KJ - KH
8166	16503	1	A3	KH	01:12:00	366	LHB	KJ - KH
8193	16503	2	A3	KH	01:12:00	366	LHB	KJ - KH
8222	22502	1	B1	KH	00:56:00	366	LHB	HTÅ - KH
8121	23658	1	B2	KH	19:19:00	366	LHB	HOT - KH
8666	23659	1	B2	KH	19:39:00	366	LHB	HOT - KH
8187	26603	1	B3	KH	01:09:00	366	LHB	HI - KH
8168	30502	1	C1	KH	00:48:00	366	LHB	BA - KH
8110	30602	1	C1	KH	00:57:00	366	LHB	KL - KH
8201	30603	1	C1	KH	01:17:00	366	LHB	KL - KH
8202	32657	1	C2	KH	19:07:00	366	LHB	KL - KH
8127	32658	2	C2	KH	19:27:00	366	LHB	KL - KH
8163	32658	1	C2	KH	19:27:00	366	LHB	KL - KH
8125	32659	1	C2	KH	19:47:00	366	LHB	KL - KH
4122	36503	2	C3	KH	01:08:00	150	LHF	FS - KH
8192	36503	1	C3	KH	01:08:00	366	LHB	FS - KH
8130	52500	1	H	KH	00:00:00	366	LHB	FS - KH
8183	52501	1	H	KH	00:20:00	366	LHB	FS - KH
8191	52502	1	H	KH	00:40:00	366	LHB	FS - KH
8144	52571	1	H	KH	23:40:00	366	LHB	FS - KH
8197	55626	1	BX	KH	08:55:00	366	LHB	KK - KH
8194	55937	1	BX	KH	12:30:00	366	LHB	HTÅ - KH
8444	55961	1	BX	KH	20:30:00	366	LHB	HTÅ - KH
8555	55962	1	BX	KH	20:50:00	366	LHB	HTÅ - KH
8146	60602	1	E1	KH	00:41:00	366	LHB	HL - KH
8172	62657	1	E2	KH	19:11:00	366	LHB	HI - KH
4110	71057	1	F3	KH	19:15:00	150	LHF	HL - KH
4112	72003	1	F1	KH	01:05:00	150	LHF	HL - KH
8124	72003	2	F1	KH	01:05:00	366	LHB	HL - KH

En illustration af input data omkring afgående tognumre en specifik ugedag



start_tognr	Dage	KM	start_lob	start_linie	Afgang	Demand	Demand_Val	start_rute
12215	5	1764,28	KH	A1	05:14:00	336	1	KH - UND
13219	4	957,59	KH	A2	06:24:00	336	1	KH - UND
22215	3	19,402	KH	B1	05:10:00	336	1	KH - HTÅ
22216	3	19,402	KH	B1	05:30:00	336	1	KH - HTÅ
23118	3	440,546	KH	B2	06:07:00	336	1	KH - HOT
23219	3	459,948	KH	B2	06:20:00	336	1	KH - HTÅ
23220	3	459,948	KH	B2	06:40:00	336	1	KH - HTÅ
30115	2	3181,049	KH	C1	05:09:00	336	1	KH - KL
30214	4	625,74	KH	C1	04:58:00	336	1	KH - BA
30215	4	625,74	KH	C1	05:18:00	336	1	KH - BA
30216	4	598,67	KH	C1	05:38:00	336	1	KH - BA
32117	4	799,534	KH	C2	05:59:00	336	1	KH - KL
32118	1	3832,2	KH	C2	06:19:00	336	1	KH - KL
32218	3	551,76	KH	C2	06:08:00	336	1	KH - FS
32219	3	551,76	KH	C2	06:28:00	336	1	KH - FS
52117	3	813,488	KH	H	05:41:00	336	1	KH - KK
52117	3	529,642	KH	H	05:41:00	150	0,5	KH - KK
52215	4	1134,847	KH	H	05:06:00	336	1	KH - FS
52215	5	676,804	KH	H	05:06:00	150	0,5	KH - FS
52216	4	896,77	KH	H	05:26:00	336	1	KH - FS
52217	3	612,924	KH	H	05:46:00	336	1	KH - FS
56034	3	19,402	KH	BX	11:36:00	336	1	KH - HTÅ
56058	3	19,402	KH	BX	19:36:00	336	1	KH - HTÅ
56059	3	19,402	KH	BX	19:56:00	336	1	KH - HTÅ
60115	5	1753,55	KH	E1	05:05:00	336	1	KH - HI
60216	2	5149,526	KH	E1	05:22:00	336	1	KH - KJ
62118	4	1276,748	KH	E2	06:15:00	336	1	KH - HI
62218	4	1468,654	KH	E2	06:12:00	336	1	KH - KJ
62218	0	1891,416	KH	E2	06:12:00	336	1	KH - KJ
72914	5	1125,648	KH	F2	04:51:00	336	1	KH - HL
72915	2	2634,664	KH	F2	05:11:00	150	0,5	KH - HL
72915	2	2093,587	KH	F2	05:11:00	150	0,5	KH - HL

:

Bilag 2 - Opstilning af original løsning

Resultatet over den faktiske planlægning onsdag d. 25.02.2014. Her har operatøren hægtet ankomne togsæt til togløb og sammensat en plan for parkeringen:

København Depot Vest

6 Sæt 4.Gen 131		5,5 Sæt 4.Gen 132		5 Sæt 4.Gen 134		4 Sæt 4.Gen 135		2 Sæt 4.Gen 133			
8187	12215 5:14	8191	60115 5:05	8166	30214 4:58	8201	72914 4:51	8125	32117 5:59		
8110	22216 5:30	8130	30115 5:09	8193	22215 5:10	4122 4112	72915 5:11	8172 Gra-B	23118 6:07		
8168	30216 5:38	8144 8121	62218 6:12	8192	30215 5:18	8124	62118 6:15				
8183	32218 6:08	8202 SAND	23219 6:20	22502	52216 5:26	8148	32118 6:19	**2 Sæt 4.Gen 136**		**3 Sæt 4.Gen 137**	
								23659	60216 5:22	4131 8198	52215 5:06
8152	32219 6:28			8163	52217 5:46			X 8137		4110 8199	52117 5:41
8190 SAND	23220 6:40			8127	13219 6:24						

Følgende to illustrationer viser, hvordan Rangerplanen ser ud for onsdag d. 25.02.2014:

København H

25-02-2014

Rangerplan for 25-02-14 fra 13:06 - 25:17

Tid	Tognr	Vogn	Vask	Spor	Tiltog	Afg.
13:31	57940- 1	8102 LHB	AFG	136		
13:46	23141- 1	8143 LHB	AFG	137		
13:46	23141- 1	8137 LHB	ANK	136		
18:42	13556- 1	8152 LHB	ANK	131	32219- 2	06:28
	- 2	8190 LHB	ANK	131	23220- 2	06:40
19:03	13657- 1	8199 LHB	ANK	137	52117- 2	05:41
19:07	32657- 2	8202 LHB	ANK	132	23219- 2	06:20
19:11	62657- 1	8172 LHB	ANK	133	23118- 2	06:07
19:15	71057- 2	4110 LHF	ANK	137	52117- 1	05:41
19:19	23658- 2	8121 LHB	ANK	132	62218- 2	06:12
19:27	32658- 1	8163 LHB	ANK	134	52217- 2	05:46
	- 2	8127 LHB	ANK	134	13219- 1	06:24
19:36	56058- 1		AFG			
19:39	23659- 2		ANK	136	60216- 1	05:22
19:47	32659- 2	8125 LHB	ANK	133	32117- 2	05:59
19:56	56059- 1		AFG			
20:30	55961- 1		ANK			
20:50	55962- 1		ANK			

Side - 1

København H

25-02-2014

Rangerplan for 25-02-14 fra 13:06 - 25:17

Tid	Tognr	Vogn	Vask	Spor	Tiltog	Afg.
23:40	52571- 2	8144 LHB	ANK	132	62218- 1	06:12
00:00	52500- 2	8130 LHB	ANK	132	30115- 1	05:09
00:20	52501- 2	8183 LHB	ANK	131	32218- 2	06:08
00:40	52502- 2	8191 LHB	ANK	132	60115- 2	05:05
00:41	60602- 1	8146 LHB	ANK	135	32118- 1	06:19
00:48	30502- 1	8168 LHB	ANK	131	30216- 1	05:38
00:52	16502- 1	4131 LHF	ANK	137	52215- 1	05:06
	- 2	8198 LHB	ANK	137	- 2	05:06
00:56	22502- 1		ANK	134	52216- 2	05:26
00:57	30602- 1	8110 LHB	ANK	131	22216- 3	05:30
01:05	72003- 1	4112 LHF	ANK	135	72915- 2	05:11
	- 2	8124 LHB	ANK	135	62118- 1	06:15
01:08	36503- 1	4122 LHF	ANK	135	72915- 1	05:11
	- 2	8192 LHB	ANK	134	30215- 1	05:18
01:09	26603- 2	8187 LHB	ANK	131	12215- 1	05:14
01:12	16503- 1	8166 LHB	ANK	134	30214- 1	04:58
	- 2	8193 LHB	ANK	134	22215- 2	05:10
01:17	30603- 1	8201 LHB	ANK	135	72914- 1	04:51

Side - 2

Bilag 3 - Depotstruktur KH Vest

Oversigt over depotstruktur på depot Vest ved KH. Det fremgår at der ialt er 53 virtuelle enhedsdepotpladser, hvilket svarer til 53 mulige parkeringer af LHF toge. Begrænsningen for hvert depotspor angives i meter og længden af hver type enhed angives til LHF = 42,45 & LHB = 83,78.

	Depot k	1	2	3	4	5	6	7
	Depot	131	132	133	134	135	136	137
Virtuelle enhedsdepoter	**Depotplads 1**	i = 1	i = 13	i = 24	i = 28	i = 37	i = 44	i = 48
	Depotplads 2	i = 2	i = 14	i = 25	i = 29	i = 38	i = 45	i = 49
	Depotplads 3	i = 3	i = 15	i = 26	i = 30	i = 39	i = 46	i = 50
	Depotplads 4	i = 4	i = 16	i = 27	i = 31	i = 40	i = 47	i = 51
	Depotplads 5	i = 5	i = 17		i = 32	i = 41		i = 52
	Depotplads 6	i = 6	i = 18		i = 33	i = 42		i = 53
	Depotplads 7	i = 7	i = 19		i = 34	i = 43		
	Depotplads 8	i = 8	i = 20		i = 35			
	Depotplads 9	i = 9	i = 21		i = 36			
	Depotplads 10	i = 10	i = 22					
	Depotplads 11	i = 11	i = 23					
	Depotplads 12	i = 12						
Tilladte LHF		12	11	4	9	7	4	6
Tilladte LHB		6	6	2	5	4	2	3
Kapacitet (m)		**512**	**509**	**187**	**425,7**	**336**	**171,5**	**260**

Bilag 4 - Test af program

Testskema over programmer

Test nr.	Statements / Scenarie## maxnodes	nodesee	prescolver	primalin	heuristics	allcuts	anklebg*	Model 2 cpu time*	Model 2 noder	Model 2 Solution	Model 4 cpu time*	Model 4 noder	Model 4 Solution	Model 3 cpu time*	Model 3 noder	Model 3 Solution	Model 1 cpu time*	Model 1 noder	Model 1 Solution	Samlet gns. ved løsning cpu time*	Samlet gns. ved løsning noder	Samlet gns. ved løsning Solution
1	1000000	Auto	-	-	-	600	92	12.328	feasible	12	850	feasible	427	90573	feasible	5	457	feasible	134	26.052	100%	
2	1000000	Bb	-	-	-	600	92	6.207	feasible	18	1200	feasible	235	55018	feasible	645	257.883	infeasible	115	20.808	75%	70
3	1000000	Be	-	-	-	600	42	4.044	feasible	17	1197	feasible	4	165	feasible	6	382	feasible	17	1.447	100%	12.993
4	1000000	Depth	-	-	-	600	23	6.408	feasible	2	100	feasible	600	1000000	infeasible	15	4.487	feasible	13	3.665	75%	
5	1000000	Auto	-	-	-	Agg	600	36	3.241	feasible	13	952	feasible	24	1765	feasible	407	137.769	infeasible	24	1.986	75%	
6	1000000	Bb	-	-	-	Agg	600	245	31.610	feasible	269	100000	22	251	52928	feasible	80	837	feasible	192	28.458	75%	73
7	1000000	Be	-	-	-	Agg	600	25	4.398	feasible	431	100000	22	17	4728	feasible	147	52.420	feasible	63	20.515	75%	13.011
8	1000000	Depth	-	-	-	Agg	600	21	2.100	feasible	4	67	feasible	600	1000000	infeasible	397	133.489	infeasible	13	1.084	50%	
9	1000000	Auto	-	-	Agg	-	600	60	1.585	feasible	15	710	feasible	11	367	feasible	5	456	feasible	23	780	100%	
10	1000000	Bb	-	-	Agg	-	600	24	895	feasible	10	366	feasible	456	100000	infeasible	288	96.278	feasible	107	32.613	75%	53
11	1000000	Be	-	-	Agg	-	600	74	13.579	feasible	9	367	feasible	186	53164	feasible	7	606	feasible	69	16.929	100%	12.902
12	1000000	Depth	-	-	Agg	-	600	21	2.100	feasible	3	98	feasible	8	856	feasible	12	2.491	feasible	11	1.386	100%	
13	1000000	Auto	-	in use	-	-	600	15	540	feasible	600	89529	22	130	26193	feasible	5	456	feasible	50	9.063	75%	
14	1000000	Bb	-	in use	-	-	600	73	4.272	feasible	60	5952	feasible	147	29509	feasible	1703	1.000.000	infeasible	93	13.244	75%	54
15	1000000	Be	-	in use	-	-	600	146	49.350	feasible	55	5953	feasible	4	165	feasible	6	382	feasible	53	13.713	100%	10.127
16	1000000	Depth	-	in use	-	-	600	182	100.000	infeasible	270	100000	22	200	1000000	infeasible	19	4.487	feasible	19	4.487	25%	
17	1000000	Auto	Agg	-	-	-	600	92	12.385	feasible	13	850	feasible	429	90593	feasible	5	456	feasible	135	26.071	100%	
18	1000000	Bb	Agg	-	-	-	600	93	6.207	feasible	17	1196	feasible	224	55018	feasible	650	257.882	infeasible	111	20.807	75%	69
19	1000000	Be	Agg	-	-	-	600	43	4.044	feasible	17	1179	feasible	4	165	feasible	6	382	feasible	18	1.443	100%	12.966
20	1000000	Depth	Agg	-	-	-	600	22	6.045	feasible	3	99	feasible	222	1000000	infeasible	15	4.487	feasible	13	3.544	75%	
21	1000000	Auto	Agg	in use	Agg	Agg	600	500	100.000	infeasible	12	394	feasible	18	712	feasible	1371	395.331	infeasible	15	553	50%	
22	1000000	Bb	Agg	in use	Agg	Agg	600	569	100.000	infeasible	18	764	feasible	149	14184	feasible	600	106.802	infeasible	84	7.474	50%	77
23	1000000	Be	Agg	in use	Agg	Agg	600	8	309	feasible	19	763	feasible	10,69	835	feasible	96	37.173	feasible	33	9.770	100%	10.423
24	1000000	Depth	Agg	in use	Agg	Agg	600	581	100.000	infeasible	4	60	feasible	544	1000000	infeasible	346	47.731	feasible	175	23.896	50%	
varians								169	35.652	20	152	35.941	20	198	41.610	18	438	212.502	17	54	10.464		9	1.273
gns								128	23.777	83%	79	17.194	83%	204	44.872	75%	285	105.964	71%	66	12.070		66	12.070
gns. ved løsning								62	8.532	83%	16	1.156	83%	127	26.497	75%	63	14.939	71%					

Bilag 5 - Program kode

```
*************************************************************************;
*Program    : Hovedprogram.sas                                           *;
*Formål     : Hovedprogram til Afvikling af                              *;
*             1) Indlæs_data Matching                                    *;
*             2) Matching_problem                                        *;
*             3) Indlæs_data Parkering                                   *;
*             4) Parkering_problem                                       *;
*             5) Resultat behandling                                     *;
*************************************************************************;
options mprint mlogic symbolgen;

*=======================================================================;
*Step 1: Sletter tabeller i Work bibliotek;
*=======================================================================;
proc datasets lib=work kill nolist memtype=data ;quit;

*=====================================================================;
*Step 2:  Definerer sti til data tabeller;
*=====================================================================;
%let Idag   =%sysfunc(putn(%sysfunc(today()),YYMMDDN.));  %put &Idag;

%let HovedSti = C:\Users\Troels\Dropbox\Afhandling\6. Progam\Hovedprogram;
%let Sti = C:\Users\Troels\Dropbox\Afhandling\6. Progam\Hovedprogram;
%let StiOut = C:\Users\Troels\Dropbox\Afhandling\6. Progam\Hovedprogram\Output;

*=======================================================================;
*Step 3: Danner makro til at importere excel filer;
*=======================================================================;
%macro ImportXLS(Outfil,Sti,Fil,sheet);
          proc import out = &Outfil
                        datafile = "&sti\&Fil."
                        DBMS = XLSX
                        replace; Sheet="&Sheet.";
                        getnames = yes;
          run;
%mend ImportXLS;

*=======================================================================;
*Step 4: Her defineres hvilkenn dag hægtningen afvikles;
*=======================================================================;
%Let ugedag = Torsdag;  %put &ugedag;

%let KMMAX = 50000;
%let Max_Vedl = 134;
%let KMMAX_Sand = 10000;
%let Max_Sand = 134;

*=======================================================================;
*Step 5: Her fra afvikle de enkelte delprogrammer;
*=======================================================================;
```

```
%let Start_time=%sysfunc(putn(%sysfunc(datetime()),datetime22.));
*====Program 1 - Indlæsning af Data til Matching ========================;

*Afvilker programmet;
           %let tst=tst1;
           %put &tst;
           %let Program1=P1 - Indlaes Data Matching;
           %put "&Hovedsti\&Program1..sas";
           %include "&Hovedsti\&Program1..sas";


*====Program 2 - Matching Problemet      ================================;

*Afvilker programmet;
           %let Program2= P2 - Matching Problemet ;
           %put "&Hovedsti\&Program2..sas";
           %include "&Hovedsti\&Program2..sas";

*====Program 3 - Indlæsning af Data til Parkering ======================;

*Afvilker programmet - Sammensat vogne;
           %let Program3= P3 - Indlaes Data PP - Sammensat vogne;
           %put "&Hovedsti\&Program3..sas";
           %include "&Hovedsti\&Program3..sas";
/*

/*Afvilker programmet - Ej Sammensat vogne;
           %let Program3= P3 - Indlaes Data PP - Ej Sammensat vogne;
           %put "&Hovedsti\&Program3..sas";
           %include "&Hovedsti\&Program3..sas";
*/

*====Program      4      -      Indlæsning      af      Data      til      Parkering
================================;

*Afvilker programmet;
           %let Program4= P4 - Parkering Problemet;
           %put "&Hovedsti\&Program4..sas";
           %include "&Hovedsti\&Program4..sas";

*====Program 5 - Resultat Behandling ================================;

*Afvilker programmet;
           %let Program5= P5 - Resultat Behandling;
           %put "&Hovedsti\&Program5..sas";
           %include "&Hovedsti\&Program5..sas";



*=============================================================================;
%let Slut_time=%sysfunc(putn(%sysfunc(datetime()),datetime22.));

%put *--------------------------------*;
```

```
%put Starttid : &Start_time. ;
%put *---------------------------------*;

%put *---------------------------------*;
%put Sluttid : &Slut_time. ;
%put *---------------------------------*;

options nomprint nomlogic nosymbolgen;
*/

*=============================================================================;
*=============================================================================;
*=============================================================================;

*********************************************************************;
* Program     : P1 - Indlaes Data Matching.sas                      *;
* Formål      : Her indlæses og dannes de nødvendige datasæt og     *;
*               variable, som er nødvendige for Matching problemet  *;
*                                                                   *;
*********************************************************************;

*=====================================================================;
*Info: UDtrækket danner følgende vairable og gemmer dem i en makro;
*=====================================================================;
/*
- &Count_ank (Antallet af ankommende Litra, Numerisk)
- &LitraEND  (Antallet af ankommende Litra, Char, 'LitraXX')
- &Count_Afg (Antallet af afgående tognr, Numerisk)
- &TasksEND  (Antallet af afgående tognr, Char, 'TasksXX')
- &V1 &V2 &V3 &V4 &V5 &V6 &V7;  (Rækkefølgende over taske til vedligehold)
- &S1 &S2 &S3 &S4 &S5 &S6 &S7 &S8 &S9;  (Rækkefølgende over taske til Sand)
- &Min_Vedl (Den laveste km distance foruden tognr defineret som vedl-linier)
- &Min_Sand (Den laveste km distance foruden tognr defineret som Sand-linier)
- &LeSup (Den højeste værdi i LIFO)
- &LeDem (Den højeste værdi i LIFO)
*/
*=====================================================================;
*INFO : Udtrækket danner følgende brugbare Tabeller til videre behandling;
*=====================================================================;
/*
- Data_ankomst
- Data_afgang
-  Km_Distance  (Indeholder  Tasks1  -  TasksXX  i  stigende  afgangstider  og  de
dertilhørende
              km distance forbundet med de enkelte tognr - RækkeVetor)
- Km_Oversigt (Hvor lang de enkelte litra har kørt - KolonneVektor)
- Supply (indeholder de konstruerede Litra1-Litra2, og hvilken kap. de udbydder)
- Demand (indeholder de efterspørgslen for de konstruerede Task1-Taskxx)
- Lifo (Lifo matricen, med i-rækker (litra) og j-kolonner (tasks)
- Data_eftersyn (indeholder info om de togsæt der skal til efteryn) */

*=====================================================================;
*Step 0: Her definere vigtige varible for programmet:
*        - Variablene er udkommenteret, da de også defineres i Hovedprogrammet;
```

```
*=========================================================================;
*Manuelle variable;
*%let KMMAX = 50000;
*%let Max_Vedl = 134;
*%let KMMAX_Sand = 10000;
*%let Max_Sand = 134;

*=========================================================================;
*Step 1: Data Indput omkring Ankomne togsæt;
*=========================================================================;
%Let Data_ank = Tognr Ankomst KH;  %put &Data_ank;

%ImportXLS(Data_Ankomst01, &Sti, &Data_ank, Data );

%ImportXLS(Data_KM01, &Sti, &Data_ank, KM);

%ImportXLS(Data_Sand01, &Sti, &Data_ank, Sand);

*=========================================================================;
*Step 2: Data Indput omkring Afgang togsæt;
*=========================================================================;
%Let Data_afg = Tognr Afgang KH;  %put &Data_afg;

%ImportXLS(Data_Afgang01, &Sti, &Data_afg, &ugedag );

*=========================================================================;
*Step 3: Data Indput omkring omdirigeringr af togsæt;
*       - filen importeres kun såfremt, der ønskes ændringer;
*=========================================================================;
%Let Eftersyn = Data_Omdirigering_&idag;  %put &Eftersyn;

%ImportXLS(Data_Omdirigering, &Sti, &Eftersyn, Eftersyn );

*=========================================================================;
*Step 4: Starter behandlingen af data omkirng ankommende togsæt;
*=========================================================================;
proc sql;
create table Data_Ankomst01b as
select a.*
      ,b.KM
             ,c.KM as KM_Sand
from Data_Ankomst01 a
left join Data_Km01 b on a.litra=b.litra
left join Data_Sand01 c on a.litra=c.litra;
quit;

data Data_Ankomst02;
set Data_Ankomst01b;
*Ændre ankomsttider efter kl. 24;
           if substr(Ankomst,1,2)='00'
            then Tid_ank1=trim("24:" !! trim(substr(Ankomst,4,5) ));
           else if substr(Ankomst,1,2)='01'
            then Tid_ank1=trim("25:" !! trim(substr(Ankomst,4,5) ));
           else if substr(Ankomst,1,2)='02'
```

```
            then Tid_ank1=trim("26:" !! trim(substr(Ankomst,4,5) ));
           else Tid_ank1=Ankomst;
run;

Data Data_Ankomst03 Data_Ankomst_EjPark;
set Data_Ankomst02;
format Tid_ank2 Tid_ank3 time8.;
format Kapacitet commax4.2 Kapdepot commax5.2;
*Opdeler;
T1=substr(Tid_ank1,1,2);
T2=substr(Tid_ank1,4,2);
T3=substr(Tid_ank1,7,2);
*;
Tid_ank2 = input(substr(T1,1,2)||':'||substr(T2,1,2)||':'||
                          substr(T3,1,2),time8.);
Tid_ank3 = Tid_ank2-14400; *Svare til 4 timer 4x3600;
*;
if type='LHB' then Kapdepot=83.78; else Kapdepot=42.58;
if type='LHB' then Kapacitet=1; else Kapacitet=0.5;

*Fjerner indkommne tognr der ankommer inden kl. 18.00 BX-linien;
           if Tid_ank1 < '18:00:00' or Linie = 'BX'
           then Output Data_Ankomst_EjPark;
           else Output Data_Ankomst03;
Drop T1 T2 T3;
run;

*Sorter ankomsttiderne i faldende orden;
proc sort data=Data_Ankomst03 out=Data_Ankomst04;
by desending Tid_ank3 position; run;

Data Data_Ankomst05;
set Data_Ankomst04;
format Litra_obs 2.;
format Tasks $10.;
Litra_obs=_n_;
Tasks=trim(left("Togsæt " !! trim(left(Litra_obs))));
run;

*=====================================================================;
*Step 5: Danner endelige fil omkring ankommende togsæt/Litra;
*=====================================================================;
Proc sql;
create table Data_ankomst as
select a.Litra_obs
     , a.Tasks as Litra
     , a.tognr
     , a.position
     , a.linie
     , a.Litra as Litra_nr
     , a.km
     , a.km_sand
     , a.Ankomst
           , a.Tid_ank1
```

```
            , a.Tid_ank2
            , a.Tid_ank3
            , a.type
            , a.Supply
            , a.kapacitet format=commax4.2
            , a.Kapdepot format=commax5.2
            , max(a.Litra_obs) as Count
from Data_Ankomst05 a;
quit;

proc sql noprint;
select count(*)
into :Count_ank
from Data_ankomst;
quit;
%put &Count_ank;

Data _null_;
format LitraEND $8.;
LitraEND=trim(left("Togsæt" !! trim(left(&Count_ank))));
call symput('LitraEND',trim(left(LitraEND)));
run; %put &LitraEND;

*=====================================================================;
*Step 6: Starter behandlingen af data omkirng Afgående togsæt;
*=====================================================================;
Data Data_Afgang02;
set Data_Afgang01;
format Tid_afg1 time8.;
*Fjerner indkommende tognr der afgår efter kl. 07.00;
          if Afgang > '07:00:00' then delete;
afg1=substr(Afgang,1,2);
afg2=substr(Afgang,4,2);
afg3=substr(Afgang,7,2);
Tid_afg1 = input(substr(afg1,1,2)||':'||substr(afg2,1,2)||':'
                        ||substr(afg3,1,2),time8.);
*Her sættter vi Kilometer distancen til 134 for A og B linjen ;
if start_linie in ('A1','A2','A3','B1','B2','B3','BX') then Km=134;
Drop afg1 afg2 afg3;
run;

*Sorter ankomsttiderne i faldende orden;
proc sort data=Data_Afgang02 out=Data_Afgang02; by Tid_afg1 Start_tognr; run;

Data Data_Afgang03;
set Data_Afgang02;
format obs 2. Tasks $8. ;
obs=_n_;
Tasks=trim(left("Tasks " !! trim(left(Obs))));
run;

*=====================================================================;
*Step 7: Danner endelige fil omkring afgående togsæt/Litra;
*=====================================================================;
```

```
Proc sql;
create table Data_Afgang as
select a.Tasks
     , a.Start_tognr as tognr
     , a.km as Km_Distance
            , a.start_linie as Linie
     , a.Afgang
            , a.Tid_afg1
            , a.dage
            , a.Demand
            , a.Demand_val as kapacitet
            , a.Obs
from Data_Afgang03 a;
quit;

proc sql noprint;
select count(*)
into :Count_afg
from Data_afgang;
quit;
%put &Count_afg;

Data _Null_;
format TasksEND $7.;
TasksEND=trim(left("Tasks" !! trim(left(&Count_afg))));
call symput('TasksEND',trim(left(TasksEND)));
run; %put &TasksEND;


*=========================================================================;
*Step 8: I det efterfølgende defineres hvilke Tasks (tognr) som
*        hører til servicelinierne.
*         - Vedlighold (HTÅ)
*         - Sand (HTÅ & UND)
*=========================================================================;
data Service;
input Vedl $2. Sand $3.;
cards;
V1 S1
V2 S2
V3 S3
V4 S4
V5 S5
V6 S6
V7 S7
V8 S8
V9 S9
; RUN;


Data MinDist(Keep= start_tognr tasks km Ser_vedl Ser_Sand)
     Vedl01 (Keep= start_tognr obs tasks afgang)
     Sand01 (Keep= start_tognr obs tasks afgang);
set Data_Afgang03;
```

```
*Finder tasks, der ikke kan varetage service for at kunne finde den
 korteste mindste distance;
           if Start_linie in ('B1','B2','B3','BX')
     then Ser_vedl='J';
            else Ser_vedl='N';
           if Start_linie in ('A1','A2','A3','B1','B2','B3','BX')
     then Ser_Sand='J';
            else Ser_Sand='N';
          output MinDist;
*Finder og outputter de taske som kan sende et litra til vedl./sand;
           if Start_linie in ('B1','B2','B3','BX')
     then output Vedl01;
           if Start_linie in ('A1','A2','A3','B1','B2','B3','BX')
     then output Sand01;
run;

*=====================================================================;
*Step 9: Danner makrovarible over hvilke Tasks der hører til Service
*        for Vedligehold;
*=====================================================================;
Data Vedl02;
set Vedl01 nobs=nobs;
format Vedl $2.;
Obs_Ny=_N_;
Vedl=trim(left("V" !! trim(left(Obs_Ny))));
run;

proc sort data=Service; by Vedl; run;
proc sort data=Vedl02; by Vedl; run;

data Vedl03 (Drop= Sand);
merge Service (in=a)
      Vedl02 (in=b);
by Vedl;
if a;
run;

Data Vedl04;
set Vedl03;
if Vedl = 'V1' then call symput('V1',trim(left(obs)));
if Vedl = 'V2' then call symput('V2',trim(left(obs)));
if Vedl = 'V3' then call symput('V3',trim(left(obs)));
if Vedl = 'V4' then call symput('V4',trim(left(obs)));
if Vedl = 'V5' then call symput('V5',trim(left(obs)));
if Vedl = 'V6' then call symput('V6',trim(left(obs)));
if Vedl = 'V7' then call symput('V7',trim(left(obs)));
run;

%put &V1 &V2 &V3 &V4 &V5 &V6 &V7;

*=====================================================================;
*Step 10: Danner makrovarible over hvilke Tasks der hører til Service
*        for Sand;
*=====================================================================;
```

```
Data Sand02;
set Sand01 nobs=nobs;
format Sand $3.;
Obs_Ny=_N_;
Sand=trim(left("S" !! trim(left(Obs_Ny))));
run;

proc sort data=Service; by Sand; run;
proc sort data=Sand02; by Sand; run;

data Sand03 (Drop= Vedl);
merge Service (in=a)
      Sand02 (in=b);
by Sand;
if a;
run;

Data Sand04;
set Sand03;
if Sand = 'S1' then call symput('S1',trim(left(obs)));
if Sand = 'S2' then call symput('S2',trim(left(obs)));
if Sand = 'S3' then call symput('S3',trim(left(obs)));
if Sand = 'S4' then call symput('S4',trim(left(obs)));
if Sand = 'S5' then call symput('S5',trim(left(obs)));
if Sand = 'S6' then call symput('S6',trim(left(obs)));
if Sand = 'S7' then call symput('S7',trim(left(obs)));
if Sand = 'S8' then call symput('S8',trim(left(obs)));
if Sand = 'S9' then call symput('S9',trim(left(obs)));
run;
%put &S1 &S2 &S3 &S4 &S5 &S6 &S7 &S8 &S9;

*=====================================================================;
*Step 11: Finder den korteste Distance bland alle Tasks (tognr), som ikke
*        er defineret som hhv. Vedl. eller Sand;
*=====================================================================;
proc sql noprint;
select Min(a.Km) as Min
into :Min_Vedl
from MinDist a
where ser_vedl='N' and ser_Sand='N';
quit;
%put &Min_Vedl;

proc sql noprint;
select Min(a.Km) as Min
into :Min_Sand
from MinDist a
where ser_Sand='N' and ser_vedl='N';
quit;
%put &Min_Sand;

*=====================================================================;
*Step 12: Danner kolonne-vektor over udbudt kapacitet for hvert litra;
*=====================================================================;
```

```
Data Supply; set Data_ankomst; keep litra kapacitet Kapdepot; run;


*=====================================================================;
*Step 13: Danner kolonne-vektor over antal kørte kilometer for hvert
*         togsæt defineres det:
*         - tilbageværende antal km før vedligholdels
*         - Kilotemter krav
*         - Skal Litra til Vedligehold
*=====================================================================;
proc sql;
create table Km_oversigt_mid as
select a.Litra
      ,a.km
            ,(&Kmmax-a.km) as KM_overskud
            ,(&max_Vedl+&min_Vedl) as KM_Krav
            , . as Vedlhold format = 1. length=3
            , a.km_sand
            ,(&Kmmax_Sand-a.km_Sand) as KM_Sand_overskud
            ,(&max_Sand+&min_Sand) as KM_Sand_Krav
            , . as Sand format = 1. length=3
from Data_Ankomst a ;
quit;

Data Km_oversigt;
set Km_oversigt_mid;
Format Vedlhold Sand 1.;
if KM_Krav > KM_overskud then Vedlhold=1;
else Vedlhold=0;
if KM_Sand_Krav > KM_Sand_overskud then Sand=1;
else Sand=0;
run;

*=====================================================================;
*Step 14: Danner række-vektor over distance af hver enkel Task(tognr);
*=====================================================================;
Data Km_Distance_mid; set Data_afgang; keep Tasks Km_Distance; run;

proc transpose data=Km_Distance_mid out=Km_Distance (drop=_name_)
    label=Distance
    prefix=Tasks;
run;


*=====================================================================;
*Step 15: Danner række-vektor over den efterspurgte kapacitet for hver
*         enkel Task(tognr);
*=====================================================================;
Data Demand_mid; set Data_Afgang; keep Tasks kapacitet; run;

proc transpose data=Demand_mid out=Demand (drop=_name_)
    label=Kapacitet
    prefix=Tasks;
run;
```

```
*=====================================================================;
*Step 16: Danner LIFO-Matice:
*         - Antal af rækker er bestem ved antallet af indgående tognr.
*         - Antal af kolonner er bestem ved antallet af udgående tognr.
*=====================================================================;
proc fcmp;
   array Tasks[&Count_ank,&Count_afg] / nosymbols;
   do row =1 to &Count_ank;
      do col = 1 to &Count_afg;
         if row = col  then Tasks[row,  col] = 2;
                       if row < col  then Tasks[row,  col] = 2**(col-row+1);
                       if row > col  then Tasks[row,  col] = 2**(row-col+1);
      end;
   end;
   rc1 = write_array('LIFO_mid', Tasks);
quit;

*Danner en kolonner der indikerer togsæt1-togsætXX;
Data LIFO;
set LIFO_mid;
format obs 2. Litra $9. ;
obs=_n_;
Litra=trim(left("Togsæt " !! trim(left(Obs))));
Drop Obs;
run;

*=====================================================================;
*Step 17: Danner makrovairale over den største værdi i LIFO matircen;
*=====================================================================;
proc sql;
create table LIFO_Max01 as
select
       max(a.Tasks1) as Max1
            ,max(a.&TasksEND.) as Max2
from Lifo a;
quit;

Data LIFO_Max02 (drop= Max1 Max2);
set LIFO_Max01;
LeSup=Max(Max1,Max2);
LeDem=Max(Max1,Max2); run;

Data _Null_;
set LIFO_Max02;
Call symput('LeSup',LeSup);
Call symput('LeDem',LeDem);
run;

%put &LeSup; %put &LeDem;

*=====================================================================;
*Step 18: Danner overblik over eftersyn, her er to scenarier ;
*         1) Her er der ingen fikseret omdirigering
```

```
*         2) Hvis bruger har definere omdirigeringer, vil filen
*            "Data_Omdirigering" eksistere og indeholde alle togsæt
*            der skal til service.
*================================================================;
* 1. scenarie;
proc sql;
create table Data_Eftersyn01 as
select a.Litra
      ,a.tognr
            ,a.Litra_nr
      ,Vedlhold as Status_Vedlhold format = 1. length=3
            ,Sand as Status_Sand format = 1. length=3
from Data_Ankomst a
left join Km_oversigt b on a.Litra=b.Litra
; quit;

* 2. scenarie;
proc sql;
create table Data_Eftersyn01 as
select a.Litra
      ,a.tognr
            ,a.Litra_nr
      ,b.Status_Vedlhold format = 1. length=3
            ,b.Status_Sand format = 1. length=3
from Data_Ankomst a
left join Data_Omdirigering b on a.Litra_nr=b.Litra
; quit;

Data Data_Eftersyn;
set Data_Eftersyn01;
Service=0;
if Status_Vedlhold= 1 then Service=1;
if Status_Sand = 1    then Service=2;
if Status_Vedlhold= 1 and Status_Sand = 1  then Service=3;
run;

*================================================================;
*Step 19: Samler variable i en tabel;
*================================================================;
Data Variable;
Count_ank=&Count_ank;
Count_Afg=&Count_Afg;
V1=&V1; V2=&V2; V3=&V3; V4=&V4; V5=&V5; V6=&V6; V7=&V7;
S1=&S1; S2=&S2; S3=&S3; S4=&S4; S5=&S5; S6=&S6; S7=&S7; S8=&S8; S9=&S9;
Min_Vedl=&Min_Vedl; Min_Sand=&Min_Sand;
LeSup=&LeSup; LeDem=&LeDem;
run;



*================================================================;
*================================================================;
*================================================================;
```

```
*****************************************************************;
* Program     : P2 - Matching Problemt.sas                      *;
* Sti         :                                                 *;
* Formål      : Asigner indkommende togsæt til udgående togløb  *;
*                                                               *;
*****************************************************************;

*================================================================;
*INFO 1: Udtrækket benytter følgende Tabeller dannet i "P1 - ...";
*================================================================;
/*
- Data_afgang
- Km_Distance (Indeholder Tasks1 - TasksXX i stigende afgangstider og
               de dertilhørende km distance - RækkeVetor)
- Km_Oversigt (Info om togsæt,Km, Km_overskud, Km_Sand, Km_Sand_overskud)
- Supply (konstruerede togsæt1-togsætXX, og hvilken kapacitet og længde)
- Demand (konstruerede Task1-TaskXX, og hvilken kapacitet de efterspørger)
- Lifo (Lifo matricen, med i-rækker (togsæt) og j-kolonner (tasks)
*/

*================================================================;
*Info 2: Udtrækket benytter følgende Variable dannet i "P0" og "p1";
*================================================================;
/*
- &Count_Afg (Antallet af afgående tognr, Numerisk)
- &V1 &V2 &V3 &V4 &V5 &V6 &V7;  (Rækkefølgende over taske til vedligehold)
- &S1 &S2 &S3 &S4 &S5 &S6 &S7 &S8 &S9;  (Rækkefølgende over taske til Sand)
- &Min_Vedl (Den laveste km distance foruden tognr defineret som vedl-linier)
- &Min_Sand (Den laveste km distance foruden tognr defineret som Sand-linier)
- &LeSup (Den højeste værdi i LIFO)
- &LeDem (Den højeste værdi i LIFO)
- &KMMAX
- &Max_Vedl
- &KMMAX_Sand
- &Max_Sand
*/

*================================================================;
*INFO 3: Udtrækket danner følgende brugbare Tabeller til videre
*        behandling og output;
*================================================================;
/*
- Matching01
- Matching_LIFO.pdf
- Matching.pdf
*/




*================================================================;
*Step 0: Her definere vigtige varible for programmet:
*        - Variablene er udkommenteret, da de også def. i Hovedprog.;
*================================================================;
```

```
*Manuelle variable;
*%let KMMAX = 50000;
*%let Max_Vedl = 134;
*%let Max_Sand = 134;

*=============================================================================;
*Step 1: Løser modellen ;
*=============================================================================;
Proc optmodel ;

Set < string > TRAINS;
Set TASKS = 1..&Count_afg;

*----- Definere parameter -------;
Number LifoCost{TRAINS, TASKS};


Number distance{TRAINS};
Number SAND_distance{TRAINS};
Number SupplyTrain{TRAINS};
Number Vedl_korr{TRAINS};
Number sand_korr{TRAINS};
Number DemandTasks{TASKS};
Number Transport{TASKS};

*----- Definere variable --------;
Var x{ TRAINS, TASKS} binary;
Var y{ TASKS} binary;
Var z{ TASKS} binary;

*----- Indlæser matricer --------;
*LIFOCOST;
read data LIFO into TRAINS = [Litra] {t in TASKS} < LifoCost[Litra,t] = col("tasks"
|| t) >;


*----- Indlæser vektorer --------;
*Lodret vektor;
read data KM_Oversigt into TRAINS = [Litra] distance=KM;
read data KM_Oversigt into TRAINS = [Litra] SAND_distance=KM_Sand;

read data Data_eftersyn into TRAINS = [Litra] vedl_korr=Status_Vedlhold;
read data Data_eftersyn into TRAINS = [Litra] sand_korr=Status_Sand;

read data Supply into TRAINS = [Litra] SupplyTrain=Kapacitet;

*Vandret vektor;
Read data Demand Into {t in TASKS} < DemandTasks[t] = col("tasks" || t) >;

Read data Km_distance Into {t in TASKS} < Transport[t] = col("tasks" || t) >;

*================================================;
*   Definer objekt funktionen;
*================================================;
```

```
*Omk for hhv. Lifo, underkap og overkap;
min obj=sum{l in TRAINS, t in TASKS}
                      x[l,t]*LifoCost[l,t] +
                      sum{t in TASKS}
                                y[t]*&LeSup +
                      sum{t in TASKS}
                                z[t]*&LeDem;

*================================================;
*   Rækkebibetingelser;
*================================================;

*Hver litra kan maksimalt optræde en gang, men er ikke nødvendigt at sende i drift;
con RowCon1{l in TRAINS}: sum{t in TASKS} x[l,t] LE 1;

*km restriktion for vedligehold;
con RowCon2{l in TRAINS}:
    sum{t in TASKS} Transport[t] * x[l,t] + distance[l] LE &KMMax;

*km restriktion for sand;
con RowCon3{l in TRAINS}:
    sum{t in TASKS} Transport[t] * x[l,t] + SAND_distance[l] LE &KMMax_Sand;

*Vedligeholdelsesrestriktion med makrovariable;
con RowCon4{l in TRAINS}:
             sum{t in TASKS : t = &V1} x[l,t] +
             sum{t in TASKS : t = &V2} x[l,t] +
             sum{t in TASKS : t = &V3} x[l,t] +
             sum{t in TASKS : t = &V4} x[l,t] +
             sum{t in TASKS : t = &V5} x[l,t] +
             sum{t in TASKS : t = &V6} x[l,t] +
             sum{t in TASKS : t = &V7} x[l,t]
           GE
           if distance[l]  GE &KMMax - &Max_Vedl - &Min_Vedl then 1 else 0;

*Sand restriktion med makrovariable;
con RowCon5{l in TRAINS}:
      sum{t in TASKS : t = &S1} x[l,t] +
             sum{t in TASKS : t = &S2} x[l,t] +
             sum{t in TASKS : t = &S3} x[l,t] +
             sum{t in TASKS : t = &S4} x[l,t] +
             sum{t in TASKS : t = &S5} x[l,t] +
             sum{t in TASKS : t = &S6} x[l,t] +
             sum{t in TASKS : t = &S7} x[l,t] +
             sum{t in TASKS : t = &S8} x[l,t] +
             sum{t in TASKS : t = &S9} x[l,t]
           GE
      if SAND_distance[l]  GE &KMMax_Sand - &Max_Sand - &Min_Sand then 1 else 0;


*Restriktion for manuelle fikseringer ;
con RowCon6{l in TRAINS}:
             sum{t in TASKS : t = &V1} x[l,t] +
             sum{t in TASKS : t = &V2} x[l,t] +
```

```
            sum{t in TASKS : t = &V3} x[l,t] +
            sum{t in TASKS : t = &V4} x[l,t] +
            sum{t in TASKS : t = &V5} x[l,t] +
            sum{t in TASKS : t = &V6} x[l,t] +
            sum{t in TASKS : t = &V7} x[l,t]
          GE Vedl_korr[l];

*Sand restriktion med makrovariable;
con RowCon7{l in TRAINS}:
      sum{t in TASKS : t = &S1} x[l,t] +
            sum{t in TASKS : t = &S2} x[l,t] +
            sum{t in TASKS : t = &S3} x[l,t] +
            sum{t in TASKS : t = &S4} x[l,t] +
            sum{t in TASKS : t = &S5} x[l,t] +
            sum{t in TASKS : t = &S6} x[l,t] +
            sum{t in TASKS : t = &S7} x[l,t] +
            sum{t in TASKS : t = &S8} x[l,t] +
            sum{t in TASKS : t = &S9} x[l,t]
          GE Sand_korr[l];

*================================================;
*   Søjlebibetingelser
*================================================;

* Hvert tognummer skal dækkes;
con ColCon1{t in TASKS}: sum{l in TRAINS} x[l,t] GE 1;

* Max 2 enheder der skal dække efterspørgslen;
con ColCon2{t in TASKS}: sum{l in TRAINS} x[l,t] LE 2;

* Udbud / efterspørgselsbalance;
con ColCon3{t in TASKS}:
            sum{l in TRAINS} SupplyTrain[l] * x[l,t] - z[t] * 0.5
          EQ
            DemandTasks[t] - y[t] * 0.5;

*================================================;
*   Nu løses modellen;
*         solve with milp / primalin;
*         solve with milp / presolver = aggressive
*         solve with milp / heuristics = aggressive;
*         solve with milp / allcuts=aggressive
*         solve with milp / maxnodes=10000
*         solve with milp / primalin;
*================================================;
*Problemet løses med mixed-integer linear programming (MILP);
Solve with milp / ;



*lav optimale værdier i sas datasætte ”Optimout”;
Create data Optimout_Match
From [TRAINS TASKS]
 = {l in TRAINS, t in TASKS: x[l,t] GE 1}
```

```
amount = x  cp = SupplyTrain[l];
quit;


*=========================================================================;
*Step 2: Fletter info på løsningen;
*=========================================================================;
Data Assign (drop=Tasks);
set Data_afgang;
rename Obs=Tasks;
run;

proc sort data=Assign out=Assign; by TASKS; run;
proc sort data=Optimout_Match out=Optimout_Match; by TASKS; run;

data Matching01;
   merge Assign Optimout_Match;
   by TASKS;
keep Linie Tasks Tognr Kapacitet Trains Amount cp Afgang tid_afg1 km_distance;
run;

*=========================================================================;
*Step 3: Sætter formater, og outputer derefter Resultatet i "Viewer";
*=========================================================================;
proc sql;
create table Tabel_Matching as
select a.*
     , b.Litra_nr as LitraNr
            , b.position
            , b.Litra_Obs
            , b.ankomst
            , b.tid_ank1
            , c.service format=1.
from Matching01 a
left join Data_ankomst b on a.trains=b.litra
left join Data_eftersyn c on a.trains=c.litra
order by b.Litra_Obs;
quit;


*=========================================================================;
*Step 4: Definerer formater;
*=========================================================================;
proc format;
   value xfmt
           0.5='LHF'
           1='LHB';
   value xcolor
           0.5='cyan'
           1='medium Green';
   value xService
           0='white'
           1='Yellow'
           2='Yellow'
```

```
           3='Yellow';
   value xNvn
           0='.'
           1='V'
           2='S'
           3='V & S';
run;

*=========================================================================;
*Step 5: Outputter overblik over Høgtningen ifht. de enkelte linjer;
*=========================================================================;
options orientation=landscape;
options papersize=(12.7in 9.0in);

ods listing close;
ods pdf file="&StiOut\Matching.pdf" style=Analysis compress=0 contents=No;

Proc tabulate data = Tabel_Matching ;
Title1 TUSP;
Title2 Hægtning af togsæt til togløb;
Class    Litra_Obs TRAINS TASKS Tognr Linie LitraNr Service
         / style={font=("Times New Roman",8pt,Bold)};
CLASSLEV Litra_Obs TRAINS TASKS Tognr Linie LitraNr
         / style={font=("Times New Roman",8pt,Bold)};
CLASSLEV Service
         / style={font=("Times New Roman",8pt,Bold) background=xservice.};
Var cp ;
Table      Litra_Obs  =  'Obs'  *  TRAINS  =  'Nr.'  *  LitraNr='Litra'  *  Service=
'Service'
             ,          Linie='Linie' * tognr='Tognr.'* TASKS* sum=" "*cp=''*f=xfmt.
         *{s=[font=("Times New Roman",8pt,Bold) background=xcolor.]}

            / BOX = {label='Hægtning af togsæt til tognr.'
            style={font=("Times New Roman",8pt,Bold)}} ;
format Service xNvn.;
run;

ods pdf close;
ods listing;

*=========================================================================;
*Step 6: Outputter overblik over Hægtningen ifht. LIFO tilgangen;
*=========================================================================;
options orientation=landscape;
options papersize=(12.7in 9.0in);

ods listing close;
ods pdf file="&StiOut\Matching_LIFO.pdf" style=Analysis compress=0 contents=No;

Proc tabulate data = Tabel_Matching ;
Title1 TUSP;
Title2 Hægtning af togsæt til togløb;
Class    Litra_Obs TRAINS TASKS Tognr Linie LitraNr kapacitet Service
         / style={font=("Times New Roman",8pt,Bold)};
```

```
CLASSLEV Litra_Obs TRAINS TASKS Tognr Linie LitraNr kapacitet
         / style={font=("Times New Roman",8pt,Bold)};
CLASSLEV Service
         / style={font=("Times New Roman",8pt,Bold) background=xservice.};
Var cp ;
Table      Litra_Obs   =   'Obs'   *   TRAINS   =   'Nr.'   *   LitraNr='Litra'   *   Service=
'Service'
             ,          TASKS    *    tognr='Tognr.'*    Linie='Linie'    *          sum="
"*cp=''*kapacitet=''*f=xfmt.
         *{s=[font=("Times New Roman",8pt,Bold) background=xcolor.]}
      / BOX = {label='Hægtning af togsæt til tognr.'
                              style={font=("Times New Roman",8pt,Bold)}} ;
format Service xNvn.;
run;

ods pdf close;
ods listing;

*=====================================================================;
*=====================================================================;
*=====================================================================;

*********************************************************************;
* Program     : P3 - Indlaes Data PP - Sammensatte vogne.sas         *;
* Sti         :                                                     *;
* Formål      : Her indlæses og dannes de nødvendige datasæt og     *;
*               variable, som er nødvendige for Parkeringsproblemet*;
*                                                                   *;
*********************************************************************;
*====================================================================;
*INFO 1: Udtrækket benytter følgende Tabeller dannet i "P2 - ...";
*====================================================================;
/*
Dannet i "P1 - Indlaes Data Matching";
  - Data_Ankomst
Dannet i "P2 - Matching Problemet";
  - Matching01
*/

*====================================================================;
*INFO 2: Udtrækket danner følgende brugbare Tabeller til videre behandling;
*====================================================================;
/*
- Resultat_matching
- MAXankomsttid (Maximum ankomst tid for det enkelte togsæt)
- MINankomsttid (Minimum ankomst tid for det enkelte togsæt)
- MAXafgangstid (Maximum afgangs tid for det enkelte togsæt
- MINafgangstid (Minimum afgangs tid for det enkelte togsæt
- Afgangstid    (Afgangs tid for den enkelte taks(Assign) opgave.)
- KapacitetsVek (Kapacitet til den enkelte Task(assign)opgave vil optage på depot.)
*/

*====================================================================;
*Step 1: Formatere output, så det er klar til videre analyse;
```

```
*================================================================;
proc sql;
create table Matching02 as
select a.*
      ,b.tognr as Ank_tognr
      ,b.Ankomst
      ,b.Tid_ank1
      ,b.Tid_ank2
      ,b.Tid_ank3
      ,case
       when a.Kapacitet = 1 then 83.78
       else 42.58 end as Kap format=commax5.2
      ,case
       when a.Cp = 1 then 'LHB'
       else 'LHF' end as Type
from Matching01 a
left join Data_Ankomst b on
  a.TRAINS=b.litra;
quit;

proc sort data=Matching02 out=Matching03; by tognr; run;

*================================================================;
*Step 2: For hver unikke tognr (som skal afgå) sættes der;
*       - Max/min ankomst
*       - Kapacitet
*       - Adresseret litra
*       - Vogn type
*================================================================;
data Matching04;
set Matching03;
format ank1 ank2 time8.;
by tognr;
retain Tasks1 Tasks2 Ank1 Ank2 Kap1 Kap2 Trains1 Trains2 Type1 Type2
       ank_tog1 ank_tog2;
        if first.tognr then do;
           Tasks1=Tasks;
           Ank1=Tid_ank3;
           Kap1=Kap;
           Trains1=Trains;
           Type1=Type;
           ank_tog1=ank_tognr;
        end;
        if last.tognr then do;
           Tasks2=Tasks;
           Ank2=Tid_ank3;
           Kap2=Kap;
           Trains2=Trains;
           Type2=Type;
           ank_tog2=ank_tognr;
           end;
           if last.tognr then output;
run;
```

```
*================================================================;
*Step 3: Her identificeres min og max ankomst for togkompositioner bestående af
*        bestående af 2 togsæt. Samtidig sikres det, at Kap., Train2 og Type2
*        kun er udfyldt, så fremt der skal benyttes to litra til tognr.
*================================================================;
Data Matching05;
set Matching04;
format Min_ankomst Max_ankomst Ank_Info time8.;
Min_ankomst=Min(Ank1,Ank2);
Max_ankomst=Max(Ank1,Ank2);
if Trains1=Trains2 then do;
           Tasks2=.;
           Kapdepot=Kap1;
           Trains2='';
           Type2='';
end;
else do; Kapdepot=Kap1+Kap2;
end;
if ank_tog1=ank_tog2 then do;
           Ank_Info=ank_tog1;
end;
else do; Ank_Info=.;
end;
Drop Kap1 Kap2;
run;

*================================================================;
*Step 4: Såfremt der er ankommende sammensatte togsæt, som ikke er hægtet
*        til et tognr, som kræver flere enheder, ligges de på række sammen,
*        så vi i pakering kan forsøge at parkerer sammensætningen på samme depot;
*================================================================;
proc sort data=Matching05 out=Matching05a; by Ank_Info; run;

data Matching05b;
set Matching05a;
format afg1_1 afg2_1 time8.;
by Ank_Info;
retain Tasks1_1 Tasks2_1 afg1_1 afg2_1 Kap1_1 Kap2_1 Trains1_1 Trains2_1
       Type1_1 Type2_1;

if ank_info = . then Output;

        if first.Ank_Info then do;
           if ank_info = . then ;
           else do;
            Tasks1_1=Tasks;
            afg1_1=Tid_afg1;
            Kap1_1=Kap;
            Trains1_1=Trains;
            Type1_1=Type;
           end;
        end;
        if last.Ank_Info then do;
           if ank_info = . then ;
```

```
            else do;
             Tasks2_1=Tasks;
             afg2_1=Tid_afg1;
             Kap2_1=Kap;
             Trains2_1=Trains;
             Type2_1=Type;
            end;
          end;
          if last.Ank_Info then output;
run;

*=================================================================;
*Step 5:
*=================================================================;
proc sort data=Matching05b out=Matching05c nodupkey;
by Trains1 Trains2; run;

Data Matching05d;
set Matching05c;
format Min_afgang Max_afgang time8.;
           if ank_Info=. then do;
                      Min_afgang=Tid_afg1;
                      Max_afgang=Tid_afg1;
           end; else do;
                      Min_afgang=Min(afg1_1,afg2_1);
                      Max_afgang=Max(afg1_1,afg2_1);
           end;
           if Trains1_1 not eq Trains2_1 then do;
               Tasks2=Tasks1_1;
                      Kapdepot=Kap1_1+Kap2_1;
                      Trains2=Trains1_1;
                      Type2=Type1_1;
           end;
Drop Tasks1_1 Tasks2_1 afg1_1 afg2_1 Kap1_1 Kap2_1 Trains1_1 Trains2_1
     Type1_1 Type2_1;
run;


*=================================================================;
*Step 6:
*=================================================================;
proc sort data=Matching05d out=Matching06; by Afgang Trains; run;

data Matching07;
set Matching06;
format Assign $10.;
by Afgang Trains;
retain Assign;
        if first.Trains then do
           Assign=trim(left("Assign " !! trim(left(_n_))));
        end;
        if last.Trains then output;
run;
```

```
*================================================================;
*Step 7: Opstiller Matching problemet i en endelig tabel;
*================================================================;
Proc sql;
create table Resultat_Matching as
select a.Linie
     , a.Assign
     , a.Tasks1
     , a.Tasks2
            , a.Tognr
            , a.Type1
            , a.Type2
            , a.Afgang
            , a.tid_afg1
            , a.Trains1
            , a.Trains2
            , a.Amount
            , a.cp
            , a.Kapdepot
            , a.Min_ankomst
            , a.Max_ankomst
            , a.Min_afgang
            , a.Max_afgang
from Matching07 a
order by a.Afgang;
quit;

*================================================================;
*Step 8: Danner kolonne-vektor over minimum ankomst tid for den enkelte
*        Litra tilknyttet en taks(Assign) opgave.
*        -  Vektor indeholder den additive inverse ankomsttid;
*================================================================;
Data MINankomsttid_mid;
set Resultat_Matching;
format ankomst_min time8.;
ankomst_min=(24*60*60)-Min_ankomst;
keep Assign Ankomst_min;
run;

*Data soreteres her udfra den additve inverse ankomst tid;
proc sort data=MINankomsttid_mid out=MINankomsttid; by Ankomst_min; run;


*================================================================;
*Step 9: Danner kolonne-vektor over Maximum ankomst tid for den enkelte
*        Litra tilknyttet en taks(Assign) opgave.
*        - Vektor indeholder den additive inverse ankomsttid;
*================================================================;
Data MAXankomsttid_mid;
set Resultat_Matching;
format ankomst_max time8.;
ankomst_max=(24*60*60)-Max_ankomst;
keep Assign Ankomst_max;
run;
```

```
*Data soreteres her udfra den additve inverse ankomst tid;
proc sort data=MAXankomsttid_mid out=MAXankomsttid; by Ankomst_max; run;

*================================================================;
*Step 10: Danner kolonne-vektor over Afgangs tid for den enkelte
*        taks(Assign) opgave.
*================================================================;
Data Afgangstid;
set Resultat_Matching;
keep Assign tid_afg1;
run;

Data MaxAfgangstid;
set Resultat_Matching;
keep Assign Max_afgang;
run;

Data MinAfgangstid;
set Resultat_Matching;
keep Assign Min_afgang;
run;

*================================================================;
*Step 11: Danner kolonne-vekoter over den kapacitet den enkelte Task(assign)
*        opgave vil optage på depot;
*================================================================;
Data KapacitetsVek;
set Resultat_Matching;
keep Assign Kapdepot;
run;
;

*================================================================;
*================================================================;
*================================================================;

*****************************************************************;
* Program     : P4 - Parkering.sas                              *;
* Sti         :                                                 *;
* Formål      : Asigner indkommende togsæt til parkering        *;
*                                                               *;
*****************************************************************;

option pagesize=50;
*================================================================;
*INFO 1: Udtrækket benytter Tabeller dannet i "P3 - ...";
*================================================================;
/*
- Resultat_matching
- MAXankomsttid (Maximum ankomst tid for det enkelte togsæt)
- MINankomsttid (Minimum ankomst tid for det enkelte togsæt)
- MAXafgangstid (Maximum afgangs tid for det enkelte togsæt
- MINafgangstid (Minimum afgangs tid for det enkelte togsæt
```

```
- Afgangstid    (Afgangs tid for den enkelte taks(Assign) opgave.)
- KapacitetsVek (Kapacitet til den enkelte Task(assign)opgave vil optage på depot.)
*/

*================================================================;
*INFO 2: Udtrækket danner følgende brugbare Tabeller til videre behandling;
*================================================================;
/*
- Parkering01
*/

*================================================================;
*Step 0: Her defineres tidintervallet mellem parkeringen af togsæt
*================================================================;
%let korr_afg = 600;
%let korr_ank = 600;

*================================================================;
*Step 1: Løser modellen ;
*================================================================;
Proc optmodel;

Set < string > TRAINS;
Set TASKS = 1..53;

*----- Definere parameter -------;
Number MINankomst{TRAINS};
Number MAXankomst{TRAINS};
Number MINafgang{TRAINS};
Number MAXafgang{TRAINS};
Number Kapac{TRAINS};


*----- Definere variable --------;
Var x{TRAINS, TASKS} binary;


*----- Indlæser vektorer --------;
*Lodret vektor;
read data MINankomsttid into TRAINS = [Assign] MINankomst=ankomst_min;
read data MAXankomsttid into TRAINS = [Assign] MAXankomst=ankomst_max;

read data MINafgangstid into TRAINS = [Assign] MINafgang=min_afgang;
read data MAXafgangstid into TRAINS = [Assign] MAXafgang=max_afgang;

read data kapacitetsVek into TRAINS = [Assign] Kapac=Kapdepot;

*================================================;
*   Definer objekt funktionen;
*================================================;;
max obj=sum{l in TRAINS, t in TASKS}
                                x[l,t];

*================================================;
```

```
*   rækkebibetingelser;
*================================================;
*Hver litra kan maksimalt optræde en gang, men er ikke nødvendigt at sende i drift;
con RowCon1{l in TRAINS}: sum{t in TASKS} x[l,t] <= 1;

*================================================;
*   Søjlebibetingelser
*================================================;
*Maksimalt 1 enhed pr. spot;
con ColCon1{t in TASKS}: sum{l in TRAINS} x[l,t] LE 1;

*Kapaciteten på depotspor skal overholdes;

*Depotspor 131;
con Kap131 : sum{l in TRAINS} x[l,1] * kapac[l] +
                      sum{l in TRAINS} x[l,2] * kapac[l] +
                      sum{l in TRAINS} x[l,3] * kapac[l] +
                      sum{l in TRAINS} x[l,4] * kapac[l] +
                      sum{l in TRAINS} x[l,5] * kapac[l] +
                      sum{l in TRAINS} x[l,6] * kapac[l] +
                      sum{l in TRAINS} x[l,7] * kapac[l] +
                      sum{l in TRAINS} x[l,8] * kapac[l] +
                      sum{l in TRAINS} x[l,9] * kapac[l] +
                      sum{l in TRAINS} x[l,10] * kapac[l] +
                      sum{l in TRAINS} x[l,11] * kapac[l] +
                      sum{l in TRAINS} x[l,12] * kapac[l]
           LE 512;

*Depotspor 132;
con Kap132 : sum{l in TRAINS} x[l,13] * kapac[l] +
                      sum{l in TRAINS} x[l,14] * kapac[l] +
                      sum{l in TRAINS} x[l,15] * kapac[l] +
                      sum{l in TRAINS} x[l,16] * kapac[l] +
                      sum{l in TRAINS} x[l,17] * kapac[l] +
                      sum{l in TRAINS} x[l,18] * kapac[l] +
                      sum{l in TRAINS} x[l,19] * kapac[l] +
                      sum{l in TRAINS} x[l,20] * kapac[l] +
                      sum{l in TRAINS} x[l,21] * kapac[l] +
                      sum{l in TRAINS} x[l,22] * kapac[l] +
                      sum{l in TRAINS} x[l,23] * kapac[l]
           LE 509;

*Depotspor 133;
con Kap133 : sum{l in TRAINS} x[l,24] * kapac[l] +
                      sum{l in TRAINS} x[l,25] * kapac[l] +
                      sum{l in TRAINS} x[l,26] * kapac[l] +
                      sum{l in TRAINS} x[l,27] * kapac[l]
           LE 187;


*Depotspor 134;
con Kap134 : sum{l in TRAINS} x[l,28] * kapac[l] +
                      sum{l in TRAINS} x[l,29] * kapac[l] +
                      sum{l in TRAINS} x[l,30] * kapac[l] +
```

```
                        sum{l in TRAINS} x[l,31] * kapac[l] +
                        sum{l in TRAINS} x[l,32] * kapac[l] +
                        sum{l in TRAINS} x[l,33] * kapac[l] +
                        sum{l in TRAINS} x[l,34] * kapac[l] +
                        sum{l in TRAINS} x[l,35] * kapac[l] +
                        sum{l in TRAINS} x[l,36] * kapac[l] +
            LE 425.7;

*Depotspor 135;
con Kap135 : sum{l in TRAINS} x[l,37] * kapac[l] +
                        sum{l in TRAINS} x[l,38] * kapac[l] +
                        sum{l in TRAINS} x[l,39] * kapac[l] +
                        sum{l in TRAINS} x[l,40] * kapac[l] +
                        sum{l in TRAINS} x[l,41] * kapac[l] +
                        sum{l in TRAINS} x[l,42] * kapac[l] +
                        sum{l in TRAINS} x[l,43] * kapac[l]
            LE 336;

*Depotspor 136;
con Kap136 : sum{l in TRAINS} x[l,44] * kapac[l] +
                        sum{l in TRAINS} x[l,45] * kapac[l] +
                        sum{l in TRAINS} x[l,46] * kapac[l] +
                        sum{l in TRAINS} x[l,47] * kapac[l]
            LE 171.5;

*Depotspor 137;
con Kap137 : sum{l in TRAINS} x[l,48] * kapac[l] +
                        sum{l in TRAINS} x[l,49] * kapac[l] +
                        sum{l in TRAINS} x[l,50] * kapac[l] +
                        sum{l in TRAINS} x[l,51] * kapac[l] +
                        sum{l in TRAINS} x[l,52] * kapac[l] +
                        sum{l in TRAINS} x[l,53] * kapac[l]
            LE 260;

*tidsintervallet mellem ankommne togsæt skal overholde i parkeringen;

*Depot 131;
con afg1  : sum{l in TRAINS} x[l,1] * (MINafgang[l] - &korr_afg)
                                    >= sum{l in TRAINS} x[l,2] * MAXafgang[l] ;
con afg2  : sum{l in TRAINS} x[l,2] * (MINafgang[l] - &korr_afg)
                                    >= sum{l in TRAINS} x[l,3] * MAXafgang[l] ;
con afg3  : sum{l in TRAINS} x[l,3] * (MINafgang[l] - &korr_afg)
                                    >= sum{l in TRAINS} x[l,4] * MAXafgang[l] ;
con afg4  : sum{l in TRAINS} x[l,4] * (MINafgang[l] - &korr_afg)
                                    >= sum{l in TRAINS} x[l,5] * MAXafgang[l] ;
con afg5  : sum{l in TRAINS} x[l,5] * (MINafgang[l] - &korr_afg)
                                    >= sum{l in TRAINS} x[l,6] * MAXafgang[l] ;
con afg6  : sum{l in TRAINS} x[l,6] * (MINafgang[l] - &korr_afg)
                                    >= sum{l in TRAINS} x[l,7] * MAXafgang[l] ;
con afg7  : sum{l in TRAINS} x[l,7] * (MINafgang[l] - &korr_afg)
                                    >= sum{l in TRAINS} x[l,8] * MAXafgang[l] ;
con afg8  : sum{l in TRAINS} x[l,8] * (MINafgang[l] - &korr_afg)
                                    >= sum{l in TRAINS} x[l,9] * MAXafgang[l] ;
con afg9  : sum{l in TRAINS} x[l,9] * (MINafgang[l] - &korr_afg)
```

```
                                    >= sum{l in TRAINS} x[l,10] * MAXafgang[l] ;
con afg10 : sum{l in TRAINS} x[l,10] * (MINafgang[l] - &korr_afg)
                                    >= sum{l in TRAINS} x[l,11] * MAXafgang[l] ;
con afg11 : sum{l in TRAINS} x[l,11] * (MINafgang[l] - &korr_afg)
                                    >= sum{l in TRAINS} x[l,12] * MAXafgang[l] ;

*Depot 132;
con afg13 : sum{l in TRAINS} x[l,13] * (MINafgang[l] - &korr_afg)
                                    >= sum{l in TRAINS} x[l,14] * MAXafgang[l] ;
con afg14 : sum{l in TRAINS} x[l,14] * (MINafgang[l] - &korr_afg)
                                    >= sum{l in TRAINS} x[l,15] * MAXafgang[l] ;
con afg15 : sum{l in TRAINS} x[l,15] * (MINafgang[l] - &korr_afg)
                                    >= sum{l in TRAINS} x[l,16] * MAXafgang[l] ;
con afg16 : sum{l in TRAINS} x[l,16] * (MINafgang[l] - &korr_afg)
                                    >= sum{l in TRAINS} x[l,17] * MAXafgang[l] ;
con afg17 : sum{l in TRAINS} x[l,17] * (MINafgang[l] - &korr_afg)
                                    >= sum{l in TRAINS} x[l,18] * MAXafgang[l] ;
con afg18 : sum{l in TRAINS} x[l,18] * (MINafgang[l] - &korr_afg)
                                    >= sum{l in TRAINS} x[l,19] * MAXafgang[l] ;
con afg19 : sum{l in TRAINS} x[l,19] * (MINafgang[l] - &korr_afg)
                                    >= sum{l in TRAINS} x[l,20] * MAXafgang[l] ;
con afg20 : sum{l in TRAINS} x[l,20] * (MINafgang[l] - &korr_afg)
                                    >= sum{l in TRAINS} x[l,21] * MAXafgang[l] ;
con afg21 : sum{l in TRAINS} x[l,21] * (MINafgang[l] - &korr_afg)
                                    >= sum{l in TRAINS} x[l,22] * MAXafgang[l] ;
con afg22 : sum{l in TRAINS} x[l,22] * (MINafgang[l] - &korr_afg)
                                    >= sum{l in TRAINS} x[l,23] * MAXafgang[l] ;

*Depot 133;
con afg24 : sum{l in TRAINS} x[l,24] * (MINafgang[l] - &korr_afg)
                                    >= sum{l in TRAINS} x[l,25] * MAXafgang[l] ;
con afg25 : sum{l in TRAINS} x[l,25] * (MINafgang[l] - &korr_afg)
                                    >= sum{l in TRAINS} x[l,26] * MAXafgang[l] ;
con afg26 : sum{l in TRAINS} x[l,26] * (MINafgang[l] - &korr_afg)
                                    >= sum{l in TRAINS} x[l,27] * MAXafgang[l] ;
*Depot 134;
con afg28 : sum{l in TRAINS} x[l,28] * (MINafgang[l] - &korr_afg)
                                    >= sum{l in TRAINS} x[l,29] * MAXafgang[l] ;
con afg29 : sum{l in TRAINS} x[l,29] * (MINafgang[l] - &korr_afg)
                                    >= sum{l in TRAINS} x[l,30] * MAXafgang[l] ;
con afg30 : sum{l in TRAINS} x[l,30] * (MINafgang[l] - &korr_afg)
                                    >= sum{l in TRAINS} x[l,31] * MAXafgang[l] ;
con afg31 : sum{l in TRAINS} x[l,31] * (MINafgang[l] - &korr_afg)
                                    >= sum{l in TRAINS} x[l,32] * MAXafgang[l] ;
con afg32 : sum{l in TRAINS} x[l,32] * (MINafgang[l] - &korr_afg)
                                    >= sum{l in TRAINS} x[l,33] * MAXafgang[l] ;
con afg33 : sum{l in TRAINS} x[l,33] * (MINafgang[l] - &korr_afg)
                                    >= sum{l in TRAINS} x[l,34] * MAXafgang[l] ;
con afg34 : sum{l in TRAINS} x[l,34] * (MINafgang[l] - &korr_afg)
                                    >= sum{l in TRAINS} x[l,35] * MAXafgang[l] ;
con afg35 : sum{l in TRAINS} x[l,35] * (MINafgang[l] - &korr_afg)
                                    >= sum{l in TRAINS} x[l,36] * MAXafgang[l] ;

*Depot 135;
```

```
con afg37 : sum{l in TRAINS} x[l,37] * (MINafgang[l] - &korr_afg)
                                  >= sum{l in TRAINS} x[l,38] * MAXafgang[l] ;
con afg38 : sum{l in TRAINS} x[l,38] * (MINafgang[l] - &korr_afg)
                                  >= sum{l in TRAINS} x[l,39] * MAXafgang[l] ;
con afg39 : sum{l in TRAINS} x[l,39] * (MINafgang[l] - &korr_afg)
                                  >= sum{l in TRAINS} x[l,40] * MAXafgang[l] ;
con afg40 : sum{l in TRAINS} x[l,40] * (MINafgang[l] - &korr_afg)
                                  >= sum{l in TRAINS} x[l,41] * MAXafgang[l] ;
con afg41 : sum{l in TRAINS} x[l,41] * (MINafgang[l] - &korr_afg)
                                  >= sum{l in TRAINS} x[l,42] * MAXafgang[l] ;
con afg42 : sum{l in TRAINS} x[l,42] * (MINafgang[l] - &korr_afg)
                                  >= sum{l in TRAINS} x[l,43] * MAXafgang[l] ;

*Depot 136;
con afg44 : sum{l in TRAINS} x[l,44] * (MINafgang[l] - &korr_afg)
                                  >= sum{l in TRAINS} x[l,45] * MAXafgang[l] ;
con afg45 : sum{l in TRAINS} x[l,45] * (MINafgang[l] - &korr_afg)
                                  >= sum{l in TRAINS} x[l,46] * MAXafgang[l] ;
con afg46 : sum{l in TRAINS} x[l,46] * (MINafgang[l] - &korr_afg)
                                  >= sum{l in TRAINS} x[l,47] * MAXafgang[l] ;

*Depot 137;
con afg48 : sum{l in TRAINS} x[l,48] * (MINafgang[l] - &korr_afg)
                                  >= sum{l in TRAINS} x[l,49] * MAXafgang[l] ;
con afg49 : sum{l in TRAINS} x[l,49] * (MINafgang[l] - &korr_afg)
                                  >= sum{l in TRAINS} x[l,50] * MAXafgang[l] ;
con afg50 : sum{l in TRAINS} x[l,50] * (MINafgang[l] - &korr_afg)
                                  >= sum{l in TRAINS} x[l,51] * MAXafgang[l] ;
con afg51 : sum{l in TRAINS} x[l,51] * (MINafgang[l] - &korr_afg)
                                  >= sum{l in TRAINS} x[l,52] * MAXafgang[l] ;
con afg52 : sum{l in TRAINS} x[l,52] * (MINafgang[l] - &korr_afg)
                                  >= sum{l in TRAINS} x[l,53] * MAXafgang[l] ;

*tidsintervallet mellem udgående togsæt skal overholde i parkeringen;

*Depot 131;
con ank1 : sum{l in TRAINS} x[l,1] * (MINankomst[l] - &korr_ank)
                                  >= sum{l in TRAINS} x[l,2] * MAXankomst[l] ;
con ank2 : sum{l in TRAINS} x[l,2] * (MINankomst[l] - &korr_ank)
                                  >= sum{l in TRAINS} x[l,3] * MAXankomst[l] ;
con ank3 : sum{l in TRAINS} x[l,3] * (MINankomst[l] - &korr_ank)
                                  >= sum{l in TRAINS} x[l,4] * MAXankomst[l] ;
con ank4 : sum{l in TRAINS} x[l,4] * (MINankomst[l] - &korr_ank)
                                  >= sum{l in TRAINS} x[l,5] * MAXankomst[l] ;
con ank5 : sum{l in TRAINS} x[l,5] * (MINankomst[l] - &korr_ank)
                                  >= sum{l in TRAINS} x[l,6] * MAXankomst[l] ;
con ank6 : sum{l in TRAINS} x[l,6] * (MINankomst[l] - &korr_ank)
                                  >= sum{l in TRAINS} x[l,7] * MAXankomst[l] ;
con ank7 : sum{l in TRAINS} x[l,7] * (MINankomst[l] - &korr_ank)
                                  >= sum{l in TRAINS} x[l,8] * MAXankomst[l] ;
con ank8 : sum{l in TRAINS} x[l,8] * (MINankomst[l] - &korr_ank)
                                  >= sum{l in TRAINS} x[l,9] * MAXankomst[l] ;
con ank9 : sum{l in TRAINS} x[l,9] * (MINankomst[l] - &korr_ank)
                                  >= sum{l in TRAINS} x[l,10] * MAXankomst[l] ;
```

```
con ank10 : sum{l in TRAINS} x[l,10] * (MINankomst[l] - &korr_ank)
                                  >= sum{l in TRAINS} x[l,11] * MAXankomst[l] ;
con ank11 : sum{l in TRAINS} x[l,11] * (MINankomst[l] - &korr_ank)
                                  >= sum{l in TRAINS} x[l,12] * MAXankomst[l] ;

*Depot 132;
con ank13 : sum{l in TRAINS} x[l,13] * (MINankomst[l] - &korr_ank)
                                  >= sum{l in TRAINS} x[l,14] * MAXankomst[l] ;
con ank14 : sum{l in TRAINS} x[l,14] * (MINankomst[l] - &korr_ank)
                                  >= sum{l in TRAINS} x[l,15] * MAXankomst[l] ;
con ank15 : sum{l in TRAINS} x[l,15] * (MINankomst[l] - &korr_ank)
                                  >= sum{l in TRAINS} x[l,16] * MAXankomst[l] ;
con ank16 : sum{l in TRAINS} x[l,16] * (MINankomst[l] - &korr_ank)
                                  >= sum{l in TRAINS} x[l,17] * MAXankomst[l] ;
con ank17 : sum{l in TRAINS} x[l,17] * (MINankomst[l] - &korr_ank)
                                  >= sum{l in TRAINS} x[l,18] * MAXankomst[l] ;
con ank18 : sum{l in TRAINS} x[l,18] * (MINankomst[l] - &korr_ank)
                                  >= sum{l in TRAINS} x[l,19] * MAXankomst[l] ;
con ank19 : sum{l in TRAINS} x[l,19] * (MINankomst[l] - &korr_ank)
                                  >= sum{l in TRAINS} x[l,20] * MAXankomst[l] ;
con ank20 : sum{l in TRAINS} x[l,20] * (MINankomst[l] - &korr_ank)
                                  >= sum{l in TRAINS} x[l,21] * MAXankomst[l] ;
con ank21 : sum{l in TRAINS} x[l,21] * (MINankomst[l] - &korr_ank)
                                  >= sum{l in TRAINS} x[l,22] * MAXankomst[l] ;
con ank22 : sum{l in TRAINS} x[l,22] * (MINankomst[l] - &korr_ank)
                                  >= sum{l in TRAINS} x[l,23] * MAXankomst[l] ;

*Depot 133;
con ank24 : sum{l in TRAINS} x[l,24] * (MINankomst[l] - &korr_ank)
                                  >= sum{l in TRAINS} x[l,25] * MAXankomst[l] ;
con ank25 : sum{l in TRAINS} x[l,25] * (MINankomst[l] - &korr_ank)
                                  >= sum{l in TRAINS} x[l,26] * MAXankomst[l] ;
con ank26 : sum{l in TRAINS} x[l,26] * (MINankomst[l] - &korr_ank)
                                  >= sum{l in TRAINS} x[l,27] * MAXankomst[l] ;

*Depot 134;
con ank28 : sum{l in TRAINS} x[l,28] * (MINankomst[l] - &korr_ank)
                                  >= sum{l in TRAINS} x[l,29] * MAXankomst[l] ;
con ank29 : sum{l in TRAINS} x[l,29] * (MINankomst[l] - &korr_ank)
                                  >= sum{l in TRAINS} x[l,30] * MAXankomst[l] ;
con ank30 : sum{l in TRAINS} x[l,30] * (MINankomst[l] - &korr_ank)
                                  >= sum{l in TRAINS} x[l,31] * MAXankomst[l] ;
con ank31 : sum{l in TRAINS} x[l,31] * (MINankomst[l] - &korr_ank)
                                  >= sum{l in TRAINS} x[l,32] * MAXankomst[l] ;
con ank32 : sum{l in TRAINS} x[l,32] * (MINankomst[l] - &korr_ank)
                                  >= sum{l in TRAINS} x[l,33] * MAXankomst[l] ;
con ank33 : sum{l in TRAINS} x[l,33] * (MINankomst[l] - &korr_ank)
                                  >= sum{l in TRAINS} x[l,34] * MAXankomst[l] ;
con ank34 : sum{l in TRAINS} x[l,34] * (MINankomst[l] - &korr_ank)
                                  >= sum{l in TRAINS} x[l,35] * MAXankomst[l] ;
con ank35 : sum{l in TRAINS} x[l,35] * (MINankomst[l] - &korr_ank)
                                  >= sum{l in TRAINS} x[l,36] * MAXankomst[l] ;

*Depot 135;
```

```
con ank37 : sum{l in TRAINS} x[l,37] * (MINankomst[l] - &korr_ank)
                                    >= sum{l in TRAINS} x[l,38] * MAXankomst[l] ;
con ank38 : sum{l in TRAINS} x[l,38] * (MINankomst[l] - &korr_ank)
                                    >= sum{l in TRAINS} x[l,39] * MAXankomst[l] ;
con ank39 : sum{l in TRAINS} x[l,39] * (MINankomst[l] - &korr_ank)
                                    >= sum{l in TRAINS} x[l,40] * MAXankomst[l] ;
con ank40 : sum{l in TRAINS} x[l,40] * (MINankomst[l] - &korr_ank)
                                    >= sum{l in TRAINS} x[l,41] * MAXankomst[l] ;
con ank41 : sum{l in TRAINS} x[l,41] * (MINankomst[l] - &korr_ank)
                                    >= sum{l in TRAINS} x[l,42] * MAXankomst[l] ;
con ank42 : sum{l in TRAINS} x[l,42] * (MINankomst[l] - &korr_ank)
                                    >= sum{l in TRAINS} x[l,43] * MAXankomst[l] ;

*Depot 136;
con ank44 : sum{l in TRAINS} x[l,44] * (MINankomst[l] - &korr_ank)
                                    >= sum{l in TRAINS} x[l,45] * MAXankomst[l] ;
con ank45 : sum{l in TRAINS} x[l,45] * (MINankomst[l] - &korr_ank)
                                    >= sum{l in TRAINS} x[l,46] * MAXankomst[l] ;
con ank46 : sum{l in TRAINS} x[l,46] * (MINankomst[l] - &korr_ank)
                                    >= sum{l in TRAINS} x[l,47] * MAXankomst[l] ;

*Depot 137;
con ank48 : sum{l in TRAINS} x[l,48] * (MINankomst[l] - &korr_ank)
                                    >= sum{l in TRAINS} x[l,49] * MAXankomst[l] ;
con ank49 : sum{l in TRAINS} x[l,49] * (MINankomst[l] - &korr_ank)
                                    >= sum{l in TRAINS} x[l,50] * MAXankomst[l] ;
con ank50 : sum{l in TRAINS} x[l,50] * (MINankomst[l] - &korr_ank)
                                    >= sum{l in TRAINS} x[l,51] * MAXankomst[l] ;
con ank51 : sum{l in TRAINS} x[l,51] * (MINankomst[l] - &korr_ank)
                                    >= sum{l in TRAINS} x[l,52] * MAXankomst[l] ;
con ank52 : sum{l in TRAINS} x[l,52] * (MINankomst[l] - &korr_ank)
                                    >= sum{l in TRAINS} x[l,53] * MAXankomst[l] ;

*================================================;
*   Nu løses modellen;
*          solve with milp / primalin;
*          solve with milp / presolver = aggressive
*          solve with milp / heuristics = aggressive;
*          solve with milp / allcuts=aggressive
*          solve with milp / maxnodes=10000
*          solve with milp / primalin;
*================================================;
*Problemet løses med mixed-integer linear programming (MILP);
solve with milp / maxtime=600 nodesel=bestestimate maxnodes=100000 heuristics =
aggressive;


*lav optimale værdier i sas datasætte ”Optimout”;
Create data Optimout_Park_mid
From [TRAINS TASKS]
 = {l in TRAINS, t in TASKS: x[l,t] Gt 0.9} /*sættes til 0,99 for kun at skrive
valgte variable ud*/
amount = x  ;
quit;
```

```
*Udskriver kørselsinformationer;
%put &_OROPTMODEL_;

*=============================================================================;
*Step 2: Danner Tabel over resultat;
*=============================================================================;
Proc tabulate data = Optimout_Park_mid;
*Title &_title;
Class TRAINS TASKS;
Var amount;
Table      TRAINS =' '
         , TASKS = "Depot"*N= ' '
         / BOX = 'Assigning trains to each depot';
run;

*=============================================================================;
*Step 3: Fletter info på løsningen;
*=============================================================================;
Data Optimout_Park;
set  Optimout_Park_mid;
format Assign $9.;
Assign=trains;
rename Tasks=DepotTask;
drop trains amount ;
run;

proc sort data=Optimout_Park; by Assign; run;
proc sort data=Resultat_matching; by Assign; run;

data Parkering01;
   merge Resultat_matching Optimout_Park;
   by Assign;
run;

*=============================================================================;
*=============================================================================;
*=============================================================================;

*****************************************************************;
* Program     : P5 - Resultat Behandling.sas                    *;
* Sti         :                                                 *;
* Formål      : Her Behandles resultatet fra "P4- Park..", så data *;
*               giver mening og kan outputtes                    *;
*                                                                *;
*================================================================;
*INFO 1: Udtrækket benytter følgende Tabeller dannet i "P2 - Matching Problemet";
*================================================================;
/*
Dannet i "P1 - Indlaes Data Matching";
  - Data_Ankomst
  - Data_afgang
  - KM_oversigt
  - Data_eftersyn
```

```
Dannet i "P4 - Parkering Problemet";
  - Parkering01
*/

*=====================================================================;
*Step 1: Definerer hvor de forskellige Depot_task hører til;
*=====================================================================;
Data Parkering02;
set Parkering01;
Format Depot 3.;
If DepotTask in (1,2,3,4,5,6,7,8,9,10,11,12) then Depot=131;
If DepotTask in (13,14,15,16,17,18,19,20,21,22,23) then Depot=132;
If DepotTask in (24,25,26,27) then Depot=133;
If DepotTask in (28,29,30,31,32,33,34,35,36) then Depot=134;
If DepotTask in (37,38,39,40,41,42,43) then Depot=135;
If DepotTask in (44,45,46,47) then Depot=136;
If DepotTask in (48,49,50,51,52,53) then Depot=137;
run;

*=====================================================================;
*Step 2: Tager nu gøjde for, at flere litra kan være assignet samme tognr og depot;
*=====================================================================;
Data Flere_litra01 (drop =type1 Trains1 Tasks1);
set Parkering02;
where Trains2 not eq "";
rename Tasks2=Tasks1
              Trains2=Trains1
       Type2=Type1;
run;

*=====================================================================;
*Step 3: Sammenkæder nu
*=====================================================================;
Data Parkering03 (drop =type2 Trains2 Tasks2);
set Parkering02 Flere_litra01;
rename Tasks1=Tasks
       Trains1=Trains
       Type1=Type;
run;

*=====================================================================;
*Step 4: Danner Resultatet for Vognopstilling for de indkommende litra;
*=====================================================================;
proc sql;
create table Resultat_vognopstilling as
select distinct b.Ankomst
      , b.tid_ank1
             , e.ankomst_max
             , f.ankomst_min
             , b.tognr
             , b.litra_nr as Litra
             , b.Position
             , b.km
             , b.Km_Sand
```

```
            , a.trains
            , b.type
            , "Ankomst" as St1
            , a.Depottask
            , a.Depot as Depotspor
            , c.tognr as Til_tognr
            , c.linie
            , c.Afgang
      , d.Status_Vedlhold
      , d.Status_Sand
from Data_ankomst b
left join Parkering03 a on  b.Litra=a.Trains
left join Matching01 c on  a.Trains=c.Trains
left join data_eftersyn d on  b.Litra=d.Litra
left join Maxankomsttid e on  a.Assign=e.Assign
left join Minankomsttid f on  a.Assign=f.Assign
order by a.Depottask, a.depot, b.tid_ank1;
quit;

*=====================================================================;
*Step 5: Danner Resultatet for Rangerplanen til Koeremaendene;
*=====================================================================;
proc sql;
create table RangerPlan01 as
select distinct b.Ankomst
            , b.tid_ank1
            , b.tognr
            , b.litra_nr as Litra
            , b.km
            , b.Km_Sand
            , a.trains
            , b.type
            , "Ankomst" as St1
            , a.Depottask
            , a.Depot as Depotspor
            , c.tognr as Til_tognr
            , c.linie
            , c.Afgang
            , e.Status_Vedlhold
            , d.km_overskud
            , e.Status_Sand
            , d.km_Sand_overskud
from Data_ankomst b
left join Parkering03 a on  b.Litra=a.Trains
left join Matching01 c on  a.Trains=c.Trains
left join KM_oversigt d on  b.Litra=d.Litra
left join data_eftersyn e on  b.Litra=e.Litra
order by b.tid_ank1;
quit;

Data RangerPlan02;
set RangerPlan01
    Data_ankomst_ejpark (Keep = tognr Litra ankomst tid_ank1 type km km_sand);
run;
```

```
proc sort data=RangerPlan02 out=Resultat_RangerPlan; by tid_ank1; run;

*================================================================;
*Step 6: Rapport over togsæt og serviceinfo;
*================================================================;
proc sql;
create table Data_Omdirigering as
select distinct b.Ankomst
             , b.tognr
             , b.litra_nr as Litra
             , b.km
             , b.Km_Sand
             , b.type
             , c.tognr as Til_tognr
             , c.linie
             , c.Afgang
             , e.km_distance
             , d.km_overskud
             , d.km_Sand_overskud
             , d.vedlhold as Status_Vedlhold
             , d.sand as Status_Sand
from Data_ankomst b
left join Parkering03 a on  b.Litra=a.Trains
left join Matching01 c on  a.Trains=c.Trains
left join KM_oversigt d on  b.Litra=d.Litra
left join Data_afgang e on  a.Tasks=e.Obs
left join Data_eftersyn f on  d.Litra=f.Litra
;
quit;


proc sort data=Data_Omdirigering;
by descending Status_Vedlhold descending Status_Sand
km_Sand_overskud km_overskud; run;

*================================================================;
*Step 7: Definerer makro til import af filer;
*================================================================;
%macro ExportXLS(Resultat,Sti,Fil,sheet);

proc export data= &Resultat
            OUTFILE="&sti\&Fil."
            DBMS=XLSX
            replace; Sheet="&Sheet.";
RUN;
%mend ExportXLS;



*================================================================;
*Step 8:
*================================================================;
%let Idag   =%sysfunc(putn(%sysfunc(today()),YYMMDDN.));  %put &Idag;
```

```
%ExportXLS(Resultat_vognopstilling, &StiOut, Resultat_vognopstilling_&idag,
           vognopstilling);

%ExportXLS(Resultat_rangerplan, &StiOut, Resultat_rangerplan_&idag, rangerplan);


%macro Eftersyn();
proc sql noprint;
select sum(Status_korrektion)
into :KorrEftersyn
from Data_Omdirigering;
quit;
%put &KorrEftersyn;

%if &KorrEftersyn = 0 %then %do;
%ExportXLS(Data_Omdirigering, &StiOut, Rapport_Omdirigering_&idag, eftersyn);
%end;

%else %do;
%ExportXLS(Data_Omdirigering, &StiOut, Rapport_Omdirigering_Korrektion, eftersyn);
%end;
%mend;

%Eftersyn;

*================================================================;
*Step 9:
*================================================================;
Proc sql;
create table Tabel_Vognopstilling01 as
select a.*
     , b.Service format=1.
from Resultat_vognopstilling a
left join Data_eftersyn b on  a.Litra=b.Litra_nr
order by a.Depottask, a.depotspor, a.tid_ank1;
quit;

Data Tabel_Vognopstilling02;
set Tabel_Vognopstilling01;
if depottask=. then do;
  depottask=100;
  depotspor=200;
end;
run;
*================================================================;
*step 10: Sammenkæder nu
*================================================================;
proc sort data=Tabel_Vognopstilling02; by til_tognr Depottask tid_ank1; run;

data Tabel_Vognopstilling03;
set Tabel_Vognopstilling02;
format tael 1.;
by Til_tognr;
```

```
retain  tael;
        if first.til_tognr then tael=0;
                        tael=tael+1;
run;

Data Tabel_Vognopstilling;
set Tabel_Vognopstilling03;
format TT $2.;
if tael=1 then TT='+';
if tael=2 then TT='++';
run;

proc sort data=Tabel_Vognopstilling; by Depottask depotspor tid_ank1; run;

*================================================================;
*Step 11: Sætter formater
*================================================================;

proc format;
   value xService
           1='Yellow'
           2='Yellow'
           3='Yellow';
   value xNvn
           0='.'
           1='V'
           2='S'
           3='V & S';
   value xEjPark
           200='Red'
           100='Red'
run;

*================================================================================;
*Step 12: Outputter vognopstilingen som pdf via ODS;
*================================================================================;
options orientation=PORTRAIT;
options papersize=(6.5in 7.0in);

ods listing close;
ods  pdf  file="&StiOut\Output_Vognopstilling_&tst..pdf"  style=Analysis  compress=0
contents=No ;

Proc tabulate data = Tabel_Vognopstilling format=4.; ;
Title1 TUSP;
Title2 Parkering af togsæt på depot vest;
Class    Depottask til_tognr afgang ankomst depotspor Service TT tognr
         / style={font=("Times New Roman",9pt,Bold)};
CLASSLEV  tognr til_tognr afgang ankomst
         / style={font=("Times New Roman",8pt)};
CLASSLEV depotspor Depottask
                / style={font=("Times New Roman",8pt) background=xEjPark.};
CLASSLEV Service
         / style={font=("Times New Roman",8pt) background=xservice.};
```

```
CLASSLEV TT
         / style={font=("Times New Roman",6pt,Bold)};
Var Litra ;
Table      Depottask = "Dep.nr" * tognr='Tog_ank' * ankomst='Ankomst'*til_tognr=
           "Tog_afg" * Afgang= 'Afgang' * TT='T' * Service= 'Service'
           ,  depotspor='Depotspor'  *sum='  '  *Litra='  '*{s=[font=("Times  New
           Roman",8pt)]}
           / BOX = {label='Parkering af togsæt på depot Vest' style={font=("Times
           New Roman",9pt,Bold)}} ;
format Service xNvn.;
run;

ods pdf close;
ods listing;

*=====================================================================;
*=====================================================================;
*  FØLGENDE PROGRAM ER ET ALTERNATIV TIL P3, SÅFREMT DET IKKE ER MULIGT,
*  AT FINDE EN MULIGPARKERING.
*=====================================================================;

*********************************************************************;
* Program     : P3 - Indlaes Data PP - ej Sammensatte vogne.sas     *;
* Sti         :                                                     *;
* Formål      : Her indlæses og dannes de nødvendige datasæt og     *;
*               variable, som er nødvendige for Parkeringsproblemet*;
*                                                                   *;
*********************************************************************;
*====================================================================;
*INFO 1: Udtrækket benytter følgende Tabeller dannet i "P2 - Matching Problemet";
*====================================================================;
/*
Dannet i "P1 - Indlaes Data Matching";
  - Data_Ankomst
Dannet i "P2 - Matching Problemet";
  - Matching01
*/

*====================================================================;
*INFO 2: Udtrækket danner følgende brugbare Tabeller til videre behandling;
*====================================================================;
/*
- Resultat_matching
- MAXankomsttid (Maximum ankomst tid for det enkelte togsæt)
- MINankomsttid (Minimum ankomst tid for det enkelte togsæt)
- Afgangstid    (Afgangs tid for den enkelte taks(Assign) opgave.)
- KapacitetsVek (Kapacitet til den enkelte Task(assign)opgave vil optage på depot.)
*/


*====================================================================;
*Step 1: Formatere output, så det er klar til videre analyse;
*====================================================================;
proc sql;
```

```
create table Matching02 as
select a.*
      ,b.Ankomst
             ,b.Tid_ank1
             ,b.Tid_ank2
             ,b.Tid_ank3
      ,case
           when a.Kapacitet = 1 then 2
           else 1 end as Kap
             ,case
           when a.Cp = 1 then 'LHB'
           else 'LHF' end as Type
from Matching01 a
left join Data_Ankomst b on
  a.TRAINS=b.litra;
quit;

proc sort data=Matching02 out=Matching03; by tognr; run;

*================================================================;
*Step 2: For hver unikke tognr sættes der;
*       - Max/min ankomst
*       - Kapacitet
*       - Adresseret litra
*       - Vogn type
*================================================================;
data Matching04;
set Matching03;
format ank1 ank2 afg1 afg2 time8.;
by tognr;
retain Tasks1 Tasks2 Ank1 Ank2 Kap1 Kap2 Trains1 Trains2 Type1 Type2 afg1 afg2;
        if first.tognr then do;
           Tasks1=Tasks;
           Ank1=Tid_ank3;
           Kap1=Kap;
           Trains1=Trains;
           Type1=Type;
           afg1=Tid_afg1;
        end;
        if last.tognr then do;
           Tasks2=Tasks;
           Ank2=Tid_ank3;
           Kap2=Kap;
           Trains2=Trains;
           Type2=Type;
           afg2=Tid_afg1;
        end;
        if last.tognr then output;
Drop kap;
run;

*================================================================;
*Step 3: Her identificeres min og max ankomst for togsæt bestående af
*        2 Litra. Samtidig sikre det, at Kapacitet, Train2 og Type2
```

```
*                    kun er udfyldt, så fremt der skal benyttes to litra til
tognr.
*=================================================================;
Data Matching05;
set Matching04;
format Min_ankomst Max_ankomst Min_afgang Max_afgang time8.;
Min_ankomst=Min(Ank1,Ank2);
Max_ankomst=Max(Ank1,Ank2);
Min_afgang=Min(afg1,afg2);
Max_afgang=Max(afg1,afg2);
if Trains1=Trains2 then do;
           Tasks2=.;
           Kapacitet=Kap1;
           Trains2='';
           Type2='';
end;
else do; Kapacitet=Kap1+Kap2;
end;
Drop Kap1 Kap2;
run;

*=================================================================;
*Step 4:
*=================================================================;
proc sort data=Matching05 out=Matching06; by Afgang Trains; run;

data Matching07;
set Matching06;
format Assign $10.;
by Afgang Trains;
retain Assign;
        if first.Trains then do
           Assign=trim(left("Assign " !! trim(left(_n_))));
        end;
        if last.Trains then output;
run;

*=================================================================;
*Step 5: Opstiller Matching problemet i en endelig tabel;
*=================================================================;
Proc sql;
create table Resultat_Matching as
select a.Linie
     , a.Assign
     , a.Tasks1
           , a.Tasks2
           , a.Tognr
           , a.Type1
           , a.Type2
           , a.Afgang
           , a.tid_afg1
           , a.Trains1
           , a.Trains2
           , a.Amount
```

```
            , a.cp
            , a.Kapacitet
            , a.Min_ankomst
            , a.Max_ankomst
            , a.Min_afgang
            , a.Max_afgang
from Matching07 a
order by a.Afgang;
quit;

*================================================================;
*Step 6: Danner kolonne-vektor over minimum ankomst tid for den enkelte
*        Litra tilknyttet en taks(Assign) opgave.
*        -  Vektor indeholder den additive inverse ankomsttid;
*================================================================;
Data MINankomsttid_mid;
set Resultat_Matching;
format ankomst_min time8.;
ankomst_min=(24*60*60)-Min_ankomst;
keep Assign Ankomst_min;
run;

*Data soreteres her udfra den additve inverse ankomst tid;
proc sort data=MINankomsttid_mid out=MINankomsttid; by Ankomst_min; run;


*================================================================;
*Step 7: Danner kolonne-vektor over Maximum ankomst tid for den enkelte
*        Litra tilknyttet en taks(Assign) opgave.
*        - Vektor indeholder den additive inverse ankomsttid;
*================================================================;
Data MAXankomsttid_mid;
set Resultat_Matching;
format ankomst_max time8.;
ankomst_max=(24*60*60)-Max_ankomst;
keep Assign Ankomst_max;
run;

*Data soreteres her udfra den additve inverse ankomst tid;
proc sort data=MAXankomsttid_mid out=MAXankomsttid; by Ankomst_max; run;

*================================================================;
*Step 8: Danner kolonne-vektor over Afgangs tid for den enkelte
*        taks(Assign) opgave.
*================================================================;
Data Afgangstid;
set Resultat_Matching;
keep Assign tid_afg1;
run;

*================================================================;
*Step 8: Danner kolonne-vektor over Afgangs tid for den enkelte
*        taks(Assign) opgave.
*================================================================;
```

```
Data Afgangstid;
set Resultat_Matching;
keep Assign tid_afg1;
run;

Data MaxAfgangstid;
set Resultat_Matching;
keep Assign Max_afgang;
run;

Data MinAfgangstid;
set Resultat_Matching;
keep Assign Min_afgang;
run;

*================================================================;
*Step 9: Danner kolonne-vekoter over den kapacitet den enkelte Task(assign)
*        opgave vil optage på depot;
*================================================================;
Data KapacitetsVek;
set Resultat_Matching;
keep Assign Kapacitet;
run;
;


*================================================================;
*================================================================;
*================================================================;
```