Materiellplanlegging i NSB
Rolling stock planning in Norwegian Railways

Masteroppgave

Fagområde: Anvendt økonomi og optimering

Marianne Risberg

Institutt for industriell økonomi og teknologiledelse
NTNU - Norges teknisk- naturvitenskaplige universitet

10. juni 2005

Forord

Denne oppgaven utgjør det siste semesteret på sivilingeniørstudiet i industriell økonomi og teknologiledelse ved Norges teknisk- naturvitenskapelige universitet, NTNU, i Trondheim. Masteroppgaven er skrevet innenfor fordypningsretningen "Anvendt Økonomi og Optimering".

Arbeidet med oppgaven har foregått i samarbeid med NSB Persontog ved Planavdelingen. Målsetningen har vært å utvikle en operasjonanalytisk modell for det kombinerte vedlikeholds- og materiellturneringsproblemet til NSB.

Jeg ønsker å rette en stor takk til veilederen min ved NTNU, Asgeir Tomasgard, for god hjelp og veiledning underveis i arbeidet med masteroppgaven. Jeg ønsker også å takke Marte Fodstad og Matthias Nowak ved SINTEF for hjelp og gode forslag til arbeidet mitt. Tilslutt ønsker jeg å rette en stor takk til NSB Persontog Plan, representert ved Tore Tomasgard og Kenneth Aschehoug, for en interessant oppgave, for gode innspill og for at jeg har fått kontorplass i lokalene til NSB dette semesteret.

Marianne Risberg

Trondheim, 10.juni, 2005

Sammendrag

I denne oppgaven er det utviklet en optimeringsmodell for togplanlegging for den taktiske planleggingsfasen. Hensikten med modellen er å generere en materiellturneringsplan der krav til driftspausebasert vedlikehold er inkludert. Vedlikeholdsproblemet er formulert som et flervareflytproblem. Restriksjoner som blir tatt hensyn til i modellen er materiellтilgjengelighet og vedlikeholdskrav, samtidig som det er en del krav som blir inkludert gjennom inndatasettene.

Problembeskrivelsen og datamaterialet er hentet fra NSB, og problemet som er sett på begrenser seg til lokaltogene på Østlandet. For dette problemet er det flere materielltyper, flere vedlikeholdsbaser og flere jernbanestrekninger. Vedlikeholdsproblemet ble først sett på i Bjorvand and Risberg [2004], og der ble det vist at vedlikeholdsproblemet for lokaltog Østlandet er for stort til å bli løst til optimalitet med kun en formulering i et optimeringsbasert verktøy (XpressMP). På grunn av problemets størrelse er det i denne oppgaven sett på muligheten til å løse problemet med en skreddersydd løsningsalgoritme. Benders algoritme ble foretrukket framfor Lagrangerelaksering og Dantzig-Wolfe dekomponering.

Den implementerte algoritmen har blitt sammenliknet med en løsning i XpressMP, og det varierte for de ulike testsettene hvilken av løsningsmetodene som var best. For de fleste av de mindre testsettene hadde den implementerte algoritmen bedre løsningstid. For begge løsningsmetodene var det for lang løsningstid for de største datasettene, men løsningen i Xpress fant lovlige løsninger for alle testsettene, samt optimal løsning for alle problemene der algoritmen fant optimale løsninger. For flere av datasettene genererte optimeringsmodellen en plan som brukte færre vedlikehold enn den taktiske planen til NSB.

Optimeringsmodellen kan støtte planlegging både i taktisk og strategisk fase. I taktisk fase kan modellen generere en nedre grense for antall vedlikehold, samt generere førsteutkast av planer. I strategisk fase kan modellen brukes til konsekvensanalyser og sjekke gjennomførbarhet av konseptuelle modeller.

For vedlikeholdsproblemet på Østlandet gjenstår det fremdeles mye arbeid før problemet kan løses til optimalitet. Hvis denne optimeringsmodellen skal få en praktisk fremtidig nytte må man kunne løse problemer med kortere løsningstid enn det som ble vist i denne oppgaven. Algoritmen bør implementeres slik at den kan kjøre mer effektivt, samtidig som det må sjekkes ut om det er mulig å få Benders algoritme til å konvergere raskere. Man kan redusere problemstørrelsen ved å generere lengre dagsløp (dagsløp over flere dager) eller løse et materiellturneringsproblem per lokaltogsstrekning. Det er også argumentert for at modellen kan brukes på andre vedlikeholdsproblemer i NSB enn lokaltog Østlandet.

Innhold

Tabeller

Kapittel 1

Innledning

I Norges Statsbaner (NSB) har man i dag ingen planleggingsverktøy basert på matematiske modeller for å planlegge togmateriellturneringer. Et slikt planleggingsverktøy kan generere bedre planer og føre til en kortere planleggingsprosess. Dette fører igjen til et mer fleksibelt planleggingssystem som kan generere planer basert på ulike scenarier i stedet for kun å generere en enkelt plan [Kroon and Zuidwijk, 2003]. I samarbeid med SINTEF har NSB begynt å se på muligheten til å benytte en optimeringsmodell til å støtte materiellplanleggingsprosessen. Våren 2005 har selskapet fått utviklet en prototype som ivaretar noen av stegene i prosessen for å lage materiellturneringsplaner. Dette programmet oppfyller blant annet krav til seteetterspørsel, togmateriellтilgjengelighet og hensettingskapasitet på stasjoner og genererer dagsløp[1] i en postprosessering, men ser ikke på vedlikeholdskrav for togmateriellet.

Denne masteroppgaven er skrevet i samarbeid med SINTEF Teknologi og Samfunn og NSB. Formålet er å integrere vedlikeholdsplanlegging og planlegging av turneringer for jernbanemateriell for NSB i en optimeringsmodell.

Det er ulike oppfatninger av hvor tidlig i planleggingsfasen man skal inkludere vedlikehold. I Bjorvand and Risberg [2004] ble følgende faktorer nevnt for hvor viktig det er å planlegge vedlikehold tidlig:

Plassering av vedlikeholdsbaser. Hvis de fleste togturene passerer vedlikeholdsbasen(e), er ikke planlegging av vedlikeholdet like viktig som hvis man må bruke mye posisjonskjøring for å få togene til vedlikeholdsbasen(e) [Tomasgard, 2004a, Nõu et al., 1997].

Frekvens på vedlikehold. Hvis togene ofte må inn til vedlikehold, kreves det en nøyere planlegging enn om vedlikeholdet gjennomføres sjeldent [Cordeau et al., 2001b].

Antallet vedlikeholdsbaser. Er det få vedlikeholdsbaser blir nettverket lite fleksibelt. Dette setter større krav til vedlikeholdsplanleggingen [Cordeau et al., 2001b].

Kapasitet på vedlikeholdsbasen. En vedlikeholdsbase med begrenset kapasitet gir mindre fleksibilitet og krever bedre planlegging.

[1] Et dagsløp = en kjede av etterfølgende tog som ett togsett kjører på en dag.

Reservebeholdning. Vedlikeholdsplanleggingen er i stor grad avhengig av hvor stor reservebeholdning man har av materiell. Dette fordi store deler av materiellparken er i drift ved små reservebeholdninger [Tomasgard, 2004a, Erlebach et al., 2004].

Jernbanenettet i Norge har et stjernemønster som møtes i Oslo-tunnelen mellom Oslo S og Skøyen. Dette mønsteret begrenser fleksibiliteten da ikke alle togturer passerer en vedlikeholdsbase like naturlig. I NSB er vedlikeholdsfrekvensen avhengig av en gitt kilometergrense per materiellype. Både antallet vedlikeholdsbaser og deres kapasitet er en begrenset ressurs, og generelt har selskapet en liten reservebeholdning av materiell [Tomasgard, 2004a]. Dette viser at behovet for taktisk vedlikeholdsplanlegging er til stede, og NSB ønsker derfor også å planlegge vedlikehold på taktisk nivå.

NSB har delt opp togproduksjonen i lokaltog og regiontog. Det er også bestemt hvilke togmaterielltyper som skal benyttes på de to togproduksjonene, og de har ulikt togmateriell. Det vil si at et tog kjører enten lokaltogs- eller regiontogsstrekninger, men ikke begge. Bergen, Oslo, Stavanger og Trondheim har lokaltog og lokaltogsmateriellet i en by turnerer ikke sammen med lokaltogsmateriell i de andre byene. Derfor vil hvert lokaltogsproblem kunne betraktes uavhengig uten at man får suboptimale løsninger. Jeg har valgt å se på lokaltog i Oslo/Østlandet siden det var det som var utgangspunktet i Bjorvand and Risberg [2004] og Aschehoug and Fodstad [2002]. Dette problemet har flere materielltyper, flere vedlikeholdsbaser og flere jernbanestrekninger.

I Bjorvand and Risberg [2004] ble det utviklet en modell for taktisk vedlikeholdsplanlegging som videreutvikles i denne oppgaven. Modellen kobler sammen ferdige dagsløp til en materiellturnering, og sørger for at alle vedlikeholdskrav blir oppfylt. I denne oppgaven genereres en løsningsalgoritme for å prøve å løse vedlikeholdsproblemet fra Bjorvand and Risberg [2004] mer effektivt. Dette fordi det i Bjorvand and Risberg [2004] ble konstatert av vedlikeholdproblemet formulert rett ut i et optimeringsbasert verktøy (XpressMP) hadde lang løsningstid.

I kapittel 2 presenteres NSB og togmateriellplanleggingsproblemet. I kapittel 3 presenteres litteratur på planleggingsmodeller for tog og buss. To løsningsalgoritmer vil også bli presentert her. I kapittel 4 presenteres modellen. Først blir alle restriksjoner og mål til modellen diskutert før en modellformulering over vedlikeholdsmodellen blir presentert. Deretter blir denne modellen formulert ved hjelp av løsningsalgoritmer. I dette kapitlet velges til slutt en løsningssalgoritme for vedlikeholdsproblemet. Kapittel 5 inneholder kommentarer og diksusjon av implementeringen av løsningsalgoritmen og en framstilling av testresultatene. Diskusjon av planene som modellen genererer finner man i kapittel 6. Her vil det også diskuteres hva som er videre arbeid med vedlikeholdsproblemet. Kapittel 7 inneholder en konklusjon på oppgaven. Vedlagt til oppgaven ligger en CD med modellene, samt resultater og testsett.

Kapittel 2

NSB

NSB er et konsern som er aktive innen trafikk og vedlikehold for persontog, godstog og buss, samt eiendomsutvikling. NSB AS er morselskapet i NSB-konsernet og har ansvaret for konsernets persontogvirksomhet. NSB Persontog er delt opp i avdelingene drift, materiell, plan, salg og marked i tillegg til stabsfunksjoner.

Arbeidet i denne oppgaven gjøres i samarbeid med Persontog Plan. Planavdelingens hovedoppgave, med hensyn på materiell, er å planlegge alle bevegelser med NSBs persontogmateriell i årlige og langsiktige produksjonsplaner. Det skal sikres at det er tilstrekkelig med driftspauser til å utføre alle (materiellrelaterte) aktiviteter som er nødvendige for at NSBs togproduksjon skal kunne gjennomføres med forutsigbar kvalitet tilpasset kundens behov for sikkerhet, regularitet og punktlighet [Materiell Plan, 2003].

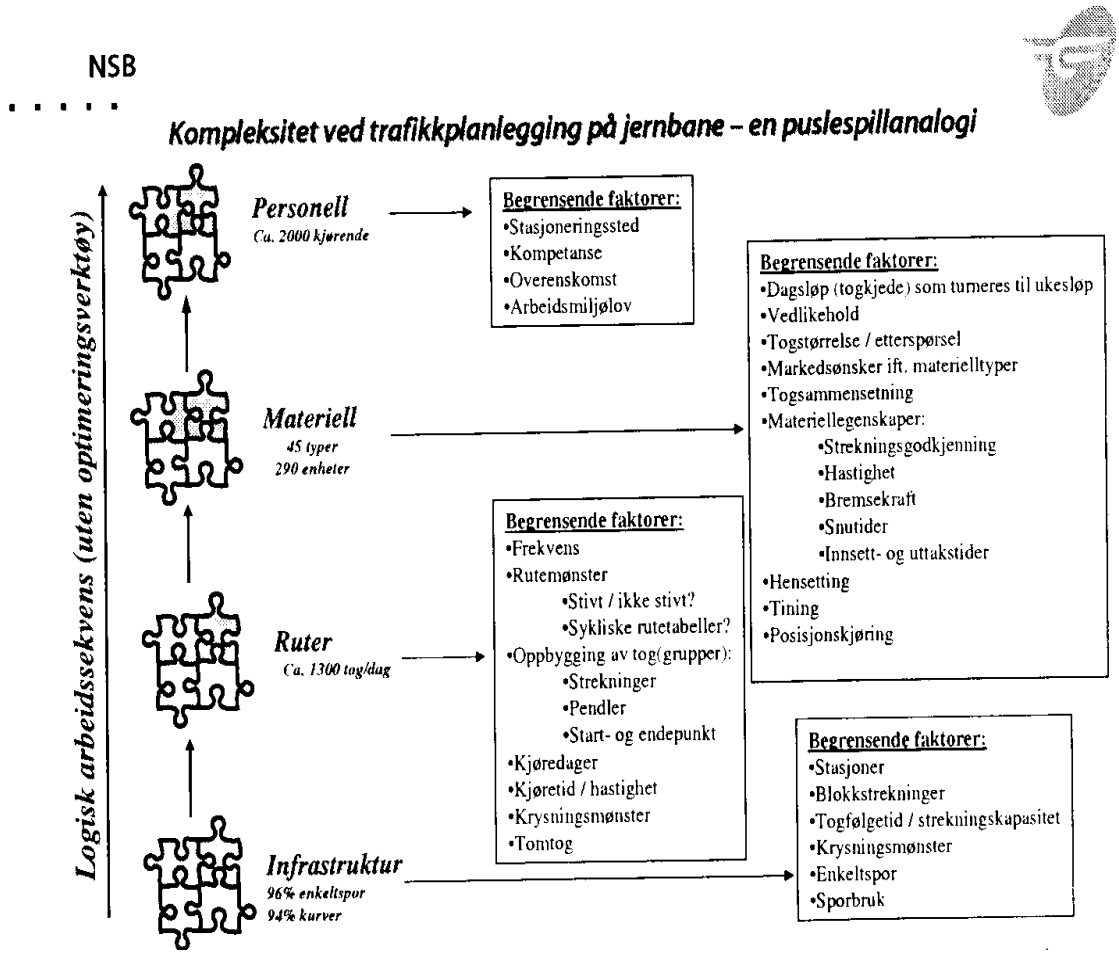
I dette kapitlet vil jeg først presentere planleggingsprosessen i NSB med fokus på materiellplanleggingsprosessen. Deretter vil lokaltogsproblemet på Østlandet bli presentert, siden det er dette problemet som modelleres og skal løses i oppgaven. Tilslutt vil materiellplanleggingsproblemet på det taktiske nivået bli presentert.

2.1 Planleggingsprosess

I NSB planlegger man langs en tidsakse og langs en funksjonsakse. Dekomponering langs funksjonsaksen kan grovt deles i ruteplanlegging, materiellplanlegging (inkludert vedlikehold) og personellplanlegging. Alle disse funksjonene henger nøye sammen og kan derfor bli planlagt under ett. På grunn av kompleksiteten i planleggingsprosessen, se figur 2.1, og manglende beslutningsstøtte foregår planleggingen mer eller mindre sekvensielt i den angitte rekkefølgen.

Dekomponering langs tidsaksen betyr at planlegging deles i flere tidsperspektiver. Forenklet kan en dele denne prosessen i fire faser, strategisk, taktisk, detalj- og operativ planlegging. Hvert av disse trinnene inneholder i større eller mindre grad de samme funksjonelle trinnene rute-, materiell- og personellplanlegging. Denne dekomponeringen av planleggingsprosessen langs funksjons- og tidsaksen er dermed en forenkling av virkeligheten [Olsson et al., 2002].

Den strategiske planleggingen er den delen av planleggingen som skjer mer enn ett år før planen skal tre i kraft. Denne planleggingen består i hovedsak av strukturelle endringer, det vil si blant annet utredning og vurdering av ulike produksjonsmodeller.

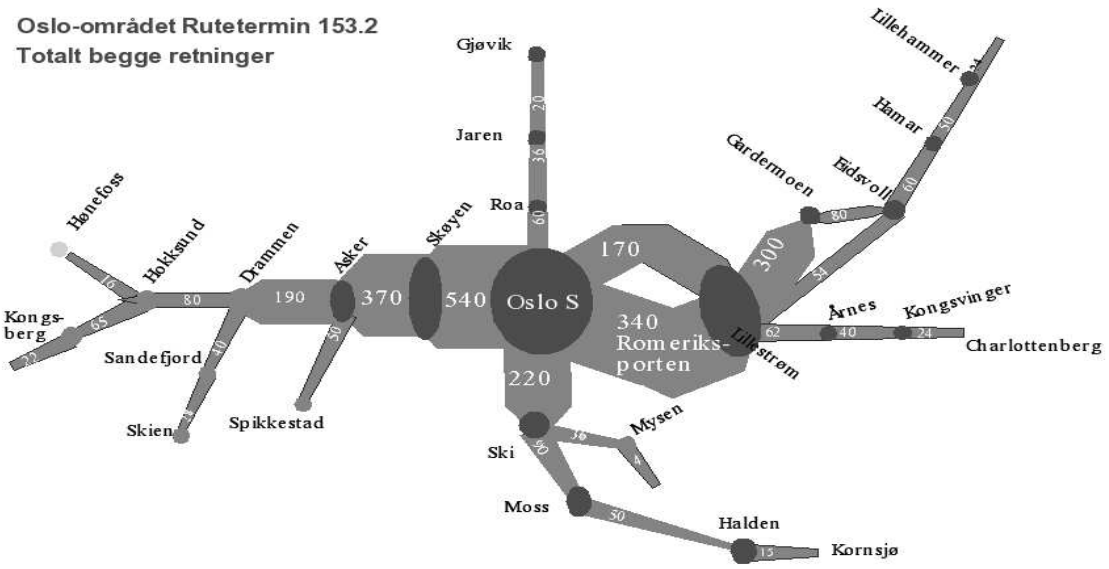


Figur 2.1: Kompleksitet ved trafikkplanlegging på jernbane.
[Tomasgard, 2004b]

Den taktiske delen av planleggingsarbeidet skjer fra et år før planen skal realiseres og frem til hovedbestillingen sendes til Jernbaneverket. Dette skjer omtrent et halvt år før den settes i drift. Som oftest vil de foreslåtte ruteplanene være en justering av foregående år, men omtrent hvert fjerde år gjøres det store endringer i ruteplanene. Detaljplanleggingen skjer når NSB har valgt og bestilt rutemodell. På dette nivået vet man hvilke avganger som skal kjøres og grovt hvilket togmateriell som skal benyttes. Dette brukes som grunnlag for utarbeidelse av personellplaner, materiellturneringsplaner og vedlikeholdsplaner. Taktisk og detaljplanlegging blir ofte sett under ett, noe som også vil bli gjort i resten av denne rapporten. Operativ planlegging er all planlegging som skjer etter at planen er satt i drift. Dette inkluderer avvikshåndtering og oppdatering av de opprinnelige planene. I praksis er det store avvik mellom de opprinnelige planene og deres realisering. Dette skyldes både eksterne forhold og forhold som NSB kan styre og påvirke selv.

2.2 Lokaltog Østlandet

I Oslo er det ca. 85 000 reisende inn og ut av byen og ca. 500 tog per virkedag, se figur 2.2. For hele NSB Persontog er det ca. 1000 tog per virkedag [Nordby, 2005].



Figur 2.2: Togtrafikken på Østlandet.
[Nordby, 2005]

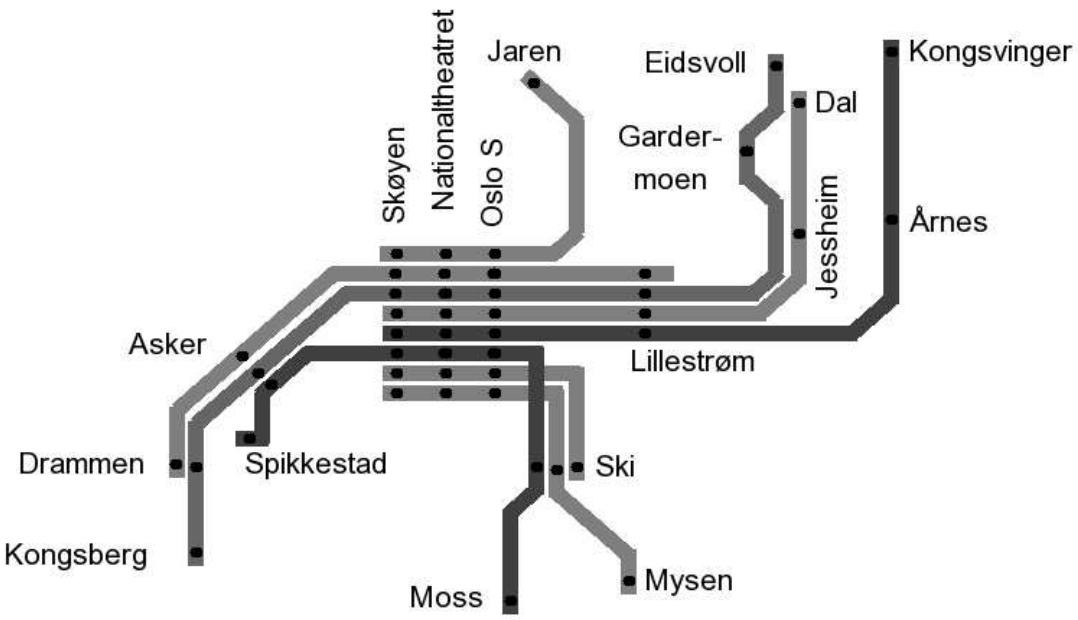
Lokaltogtrafikken på Østlandet har flere materielltyper, flere vedlikeholdsbaser og flere strekninger. En oversikt over pendlene (strekningene) finnes i figur 2.3. Regionen har to vedlikeholdsbaser for lokaltogsmateriell, Sundland ved Drammen og Filipstad ved Skøyen. Vedlikeholdet på materiellet er distanseavhengig, det vil si at et materiellindivid kan kjøre opptil et maksimalt antall kilometer mellom to vedlikehold. Materiellparken består av to hovedmaterielltyper, type 72, se figur 2.4 og 2.5 og type 69, se figur 2.6. Type 69 består av to subtyper, 2-vognsett (69-2) og 3-vognsett (69-3). Vedlikeholdskravet er likt for 69-2 og 69-3, og de kan vedlikeholdes ved begge vedlikeholdsbasene. Type 72 kan bare vedlikeholdes på Sundland. De tre materielltypene har per dags dato samme kilometergrense.

Hos NSB planlegges et vedlikehold med en varighet på fire timer. På vedlikeholdsbasene er det en kapasitetsbegrensning på hvor mange tog som kan bli vedlikeholdt samtidig. Arbeidet på vedlikeholdsbasene foregår i skift, og et tog må vedlikeholdes innenfor tidsrammen på et bestemt skift [Tomasgard, 2004a]. Dermed vil et vedlikehold bli gjennomført av samme personell. På Sundland kan togsettene vedlikeholdes hele døgnet på i alt tre skift, mens det er to skift i døgnet på Filipstad, med færre skift i helgene.

Reservebeholdningen av materiellet til NSB er liten, på 2-5%. Det er bare for type 69 de har en reserve på ca 10%. [Tomasgard, 2004a].

I lokaltogtrafikken på Østlandet brukes 10 togsett av type 69-2, 43 togsett av type 69-3 og 15 togsett av type 72 for å kjøre rutetilbudet i grunnplan 153.2[1].

[1]Grunnplanen fra januar 2005 til juni 2005. Dette har senere blitt endret til 12 togsett av type 69-2



Figur 2.3: Lokaltogsstrekningene på Østlandet.
[NSB]

2.3 Materiellplanleggingsproblemet

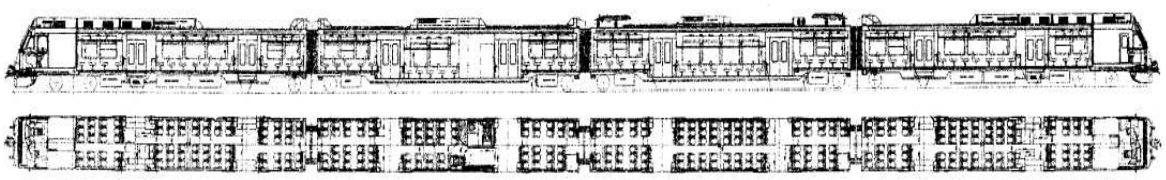
Materiellplanleggingen på det taktiske nivået starter med å ta utgangspunkt i en ruteplan. Denne ruteplanen innholder alle togturene, med avgangs- og ankomsttider og avgangs- og ankomststeder, som skal kjøres. Oppgaven til materiellplanleggeren er å tildele materiell til disse turene. Dette gjøres ved å sette opp gyldige materiellturneringer. For at en turnering skal være gyldig må alle restriksjoner til materiellplanleggingsproblemet bli oppfylt. Her vil disse restriksjonene bli beskrevet. Målene en materiellplanlegger ønsker å oppfylle vil bli beskrevet deretter.

2.3.1 Restriksjoner

Ved materiellplanlegging for tog er det følgende restriksjoner som det må tas hensyn til:

Vedlikehold. For hver materiellype er det gitt et maksimalt antall kilometer som er tillatt å kjøre mellom to vedlikehold. Denne grensen må ikke overskrides. For å gjennomføre et vedlikehold kreves en tilstrekkelig lang driftspause. I tillegg må det være ledig tid og ressurser på vedlikeholdsbasen for å få gjennomført vedlikeholdet.

og 41 togsett av type 69-3, men i denne oppgaven har jeg brukt den opprinnelige planen når jeg har generert datasett.



Figur 2.4: Skjematisk tegning av materielltype 72.
[NSB]

Materielltilgjengelighet. NSB Plan har et gitt antall enheter av hver materielltype tilgjengelig for produksjon, og dette antallet kan ikke overskrides i turneringen.

Toglengde. Det er en maksimalgrense for hvor langt et tog kan være. Denne grensen avhenger av lengden på perrongene og av tekniske begrensninger.

Skjøting og deling. Det er kun tillatt å sette sammen turer som inneholder tilstrekkelig tid til skjøting og deling der dette er nødvendig. I tillegg må infrastrukturen på stasjonene være slik at det er mulig og tillatt å gjennomføre dette. Det eksisterer et tidsstudie per materielltype, siden tiden man bruker på disse operasjonene er avhengig av materielltypen.

Snutid. Hver sekvens må inneholde tilstrekkelig tid til snuing mellom tog. Denne tiden kan være avhengig både av materielltype og infrastrukturen på snustedet.

Hensettingskapasitet ved stasjonene Ved enkelte av stasjonene er det en maksimal grense for hvor mange tog det er lov til å hensette både på dagen og på natten. Ved andre stasjoner er det ikke tillatt å hensette tog i det hele tatt. Det er Jernbaneverket som forvalter regelverket, og dermed bestemmer hva kapasiteten er på hver stasjon.

Sportilgang ved tomtogskjøring. Ved tomtogskjøring/posisjonskjøring må det sjekkes om det er ledig sportilgang innenfor det gitte tidsintervallet for denne kjøringen, og disse må også bestilles fra Jernbaneverket.

Dekke etterspørselen på alle turene. Turene skal i så stor grad som mulig dekke etterspørselen fra markedet. Etterspørselen er gitt som antall togseter per tur. Setekapasiteten som hver tur skal dekke bestemmes av materiellplanlegger/kundeplanlegger. I det taktiske materiellplanleggingsproblemet vil det være grovt bestemt hvilken materielltype som skal benyttes på de ulike togturene.

2.3.2 Mål

Alle restriksjonene er ufravikelige og må oppfylles. I tillegg er det en rekke mål som materiellplanleggeren ønsker å oppfylle.

Figur 2.5: Materielltype 72.
[NorskJernbaneklubb]

Rene pendler. Dette innebærer at turneringen for et tog i størst mulig grad skal inneholde turer som går i en og samme pendel (strekning). Dette for å gjøre materiellplanleggingen mest mulig robust og oversiktlig i forhold til operativ gjennomføring og til replanlegging. Hvis det oppstår problemer på en pendel (strekning), vil dette ha minimal konsekvens for de andre pendlene så lenge ikke materiellet fra flere pendler turnerer sammen.

Sykliske turneringer. Det er ønskelig at hvert tog etter en bestemt tid kommer tilbake til utgangspunktet og derfra gjentar turneringen. Dette øker oversikten og forutsigbarheten. Av samme grunn er det også ønskelig at syklene skal være så korte som mulig. Lengden på en sykel er antall uker før man er tilbake til utgangspunktet.

Personellkostnader. Tomtogskjøring, skjøting og deling, parkering og tilsyn av materiell krever personell, og man ønsker så lave personellkostnader som mulig.

Vedlikeholdskostnader. Vedlikeholdskostnadene avhenger av antall kilometer som blir tilbakelagt, og det er ønskelig å holde disse på et minimum. Det betyr at man ønsker en god kilometerutnyttelse. I tillegg er vedlikehold på dagen billigere enn vedlikehold på natten.

Kapitalkostnader. Den største kapitalkostnaden har man ved kapitalbinding i tog-

Figur 2.6: Materielltype 69.
[NorskJernbaneklubb]

materiellet. Derfor er det ønskelig med en materiellturnering som krever så lite materiell som mulig. For nytt materiell er det ønskelig med så høy utnyttelse som mulig på grunn av avkastning på anvendt kapital.

LIFO ved stasjonene. På noen stasjoner er det begrenset med hensettingskapasitet, eller begrenset med spor. For at kjøringene i følge planen skal foregå smidigst mulig, er det på noen stasjoner ønskelig at det toget det som kommer sist inn på kvelden kjører først ut igjen på morgenen (LIFO - last inn first out) siden flere sett blir hensatt på samme spor. Ved å ikke ta hensyn til rekkefølge på stasjoner fører det mest sannsynlig til økte personellkostnader.

Etterfølgende dagsløpsnummer. Hvert dagsløp på en dag har et nummer. Siden rutetabellen er lik mandag til fredag, er det like dagsløp mandag til fredag også. Disse dagsløpene har samme nummer. For å forenkle personellplanleggingen og planleggingen i andre avdelinger er det ønskelig at de samme dagsløpsnumrene følger etter hverandre. Det vil si at hvis man kobler sammen dagsløp 720001 mandag til dagsløp 720002 tirsdag, ønsker man å koble sammen dagsløp 720001 tirsdag til dagsløp 720002 onsdag. Etterfølgende dagsløpsnummer øker forutsigbarhet i operativ planlegging samt for driftsoperativt senter (DROPS).

Lik kjørelengde. Det er ønskelig at hvert sett i gjennomsnitt har kjørt like langt slik at slitasjen på settene er lik. Dette gjøres det ikke noe med på det taktiske planleg-

gingsnivået for lokaltog Østlandet. Ved å lage sykliske turneringer kjører togsettene ulike dagsløp uke for uke, og dette er med på å gi lik slitasje for settene. For enkelte spesielle sett, lages det egne turneringer, som kjører de samme dagsløpene uke for uke.

2.3.3 Mål for vedlikeholdsplanlegging

Vedlikeholdet blir planlagt inn i materiellturneringsplanen etter at man har dannet dagsløp av alle togturene. Et dagsløp er en sekvens av togturer et togsett kjører på en dag. Når disse dagsløpene settes sammen til ukesløp og til en hel materiellturneringsplan tas vedlikeholdskrav hensyn til. Når man bestemmer hvordan vedlikeholdet skal planlegges inn i materiellturneringsplanen er det i hovedsak disse 5 målene man retter seg etter. Materiellplanleggeren har ikke fått en absolutt prioritering på hvilke mål som skal prioriteres før andre mål, og de ulike materiellplanleggerne vektlegger derfor følgende mål forskjellig.

- God kilometerutnyttelse.
- Vedlikehold på dagtid.
- Etterfølgende dagsløpsnummer, lik overgang mellom dager.
- Ikke for lange sammenhengende sykler.
- LIFO ved enkelte stasjoner.

Kapittel 3

Litteratur og teori

I dette kapitlet vil jeg presentere litteratur på området materiellplanlegging og vedlikeholdsplanlegging. Jeg vil kommentere hvilke løsningsstrategier som er brukt for å løse noen planleggingsproblemer for tog og buss. Jeg har lagt mest vekt på de modellene som inkluderer vedlikehold og andre aspekter til materiellplanleggingsproblemet til NSB, samt de artiklene som har benyttet interessante løsningsstrategier for å løse problemene.

Deretter vil teori på løsningsprosedyrene Lagrangerelaksering og Benders dekomponering bli presentert. Disse løsningsalgoritmene ble presentert i Bjorvand and Risberg [2004] sammen med blant annet Dantzig-Wolfe dekomponering. Jeg har valgt å inkludere de to løsningsalgoritmene i dette arbeidet for å gi en grundigere forklaring på enkelte elementer. Jeg anser disse to, sammen med Dantig- Wolfe dekomponering, for alle å være aktuelle til å løse NSB sitt vedlikeholdsproblem. Ut i fra tabell 3.1 ser man at mange av problemene har blitt forsøkt løst med en av disse løsningsprosedyrene, eller i kombinasjon med en eller flere av disse.

3.1 Litteratur

En god oversikt over optimeringsmodeller som tidligere er gjort på området ruteplanlegging og materiellplanlegging for tog finnes i Cordeau et al. [1998]. Den mest vanlige tilnærmingen til togtransportsystemet er at det blir presentert som et nettverk der nodene representerer stasjoner og der kantene representerer sporene der tog kjører passasjerer eller frakt [Cordeau et al., 1998]. Dette underbygges også av det jeg har kommet over av planleggingsmodeller i arbeidet med NSBs planleggingsproblem og mange av de er presentert i tabell 3.1. I hovedsak er det modeller for togplanlegging i tabell 3.1, men jeg har også sett på noen modeller for *Multi-Depot Vehicle Scheduling Problem (MDVSP)* siden disse problemene likner litt på togplanleggingsproblemer og man har kommet langt i å løse disse problemene.

Grunnleggende antakelser til MDVSP er at hver tur blir kjørt av et kjøretøy, hvert depot har et begrenset antall kjøretøy og hvert kjøretøy returnerer periodisk til det samme depotet som det startet fra. For MDVSP og NSBs problem skal det tilordnes kjøretøy til bestemte ruter. En av hovedforskjellene er at det for hver tur kan kjøre flere togsett i NSBs problem i tillegg til at vedlikeholdskrav må bli møtt.

Löbel [1997b] benytter seg av flere ulike løsningsteknikker for å lage en løsningsalgoritme som egner seg for å løse MDVSP. Det er sett på problemer som inneholder opptil 49 depoter, ca. 25 tusen rutebestemte turer og omtrent 70 millioner heltallsvariable. For å løse dette problemet har de konstruert en egen løsningsalgoritme. Den består blant annet av elementene Lagrangerelaksering, heuristikker, lineærrelaksering, Branch & Cut samt kolonnegenerering og lineærløsningsprogrammet CPLEX. En velvalgt kombinasjon av disse metodene viste seg at kunne løse (nesten) alle problemene av praktisk interesse i akseptable kjøretider. I følge Banihashemi and Haghani er arbeidet til Löbel den mest suksessfulle tilnærmingen til å løse MDVSP.

3.1.1 Driftspausebasert vedlikehold

Hvordan man har angrepet problemer med driftspausebasert vedlikehold er interessant å se nærmere på fordi dette er hovedfokuset for planleggingsmodellen som skal utvikles i dette arbeidet. Driftspausebasert vedlikehold er vedlikehold som ligger inne i planen og er i driftspausene i materiellturneringen. To hovedangrepsmåter er sett i litteraturen for å inkludere vedlikehold, distanseavhengig eller tidsavhengig vedlikehold.

Distanseavhengig vedlikehold

Distanseavhengig vedlikehold betyr at togmateriellet kan kjøre et maksimalt antall kilometer mellom to vedlikehold. Eksempler på distanseavhengig vedlikehold finnes i Nõu et al. [1997].

Nõu et al. [1997] ser på problemet å tilordne lokomotiv til tog og det er mulig å tilordne flere lokomotiv til hvert tog. De ser på tilordning på et ukentlig problem, med en tilleggsrestriksjon at de ønsker sykliske restriksjoner. Problemet blir modellert som et flervareflyt i nettverksproblem med siderestriksjoner. De bruker et Branch & Bound skjema og bruker Dantzig-Wolfe i hver node i treet, og det blir generert et Dantzig-Wolfe subproblem per materielltype. Videre sier de at det går an å relaksere vedlikeholdskravet og heller planlegge vedlikeholdet i ettertid. Dette forutsetter at vedlikeholdsbasene er strategisk plassert i jernbanenettet og at jernbanenettet har et hensiktsmessig mønster, siden de fleste syklene da vil inneholde muligheter for vedlikehold. Hvis vedlikeholdskravet blir relaksert trenger man kun å løse $|K|$ (K = hovedmaterielltyper) subproblemer i hver iterasjon sammenliknet med $|L|$ (L = submaterielltyper) når vedlikeholdsrestriksjoner er inkludert. I deres modell var $|K| = 4$ og $|L| \approx 150$ så problemet ble betraktelig mindre ved å relaksere vedlikeholdskravet.

En av konklusjonene til Nõu et al. [1997] er at fra et modelleringssynspunkt vil det være mer hensiktsmessig å relaksere vedlikeholdsrestriksjonen fra planleggingsproblemet, siden det er bedre å ta hensyn til vedlikehold når den initielle statusen er kjent for alle lokomotiver, som for eksempel i det operasjonelle lokomotivplanleggingsproblemet.

Tidsavhengig vedlikehold

Tidsavhengig vedlikehold er vedlikehold som gjennomføres innenfor et maksimalt antall dager mellom to vedlikehold. Dette kan brukes når slitasjen på materiellet er ganske lik, slik at man i snitt trenger vedlikehold etter like mange dager for alle togsettene.

Tabell 3.1: Oversikt over noen planleggingsmodeller

Artikkel	**Plan. fase**	**Problem type**	**Modell type**	**Løsning**	**VLH**
Rasmussen [2004]	taktisk	tilordne lok+vogn	flervareflyt	Bender	nei
Erlebach et al. [2004]	taktisk	tilordne tog	envareflyt	approksimasjonsalgoritme	ja
Peeters and Kroon [2003]	taktisk	Rolling stock circ.	transition graf	Branch and price, D-W	nei[a]
Abbink et al. [2003]	taktisk	kapasitetallokering		CPLEX	nei
Alfieri et al. [2003]	taktisk	tilordne vogner	flervareflyt[d]	hierarkisk dekomponering	nei[a]
Lentink et al. [2003]	taktisk	TUSP	flere modeller	dekomponering	nei
Freling et al. [2002]	taktisk	TUSP	IP	kolonnegenerering	nei
Lingaya et al. [2002]	operasjonell	OCAP[c]	envareflyt	heuristikk	ja
Cordeau et al. [2001a]	taktisk	lok+vogn tilordning	flervareflyt	B&B og kolonnegenerering	ja
Cordeau et al. [2001b]	operasjonell	lok+vogn tilordning	flervareflyt	B&B, LP-relaksering, Bender	ja
Cordeau et al. [2000]	taktisk	lok+vogn tilordning	flervareflyt	Bender	nei
Ziarati et al. [1999]	operasjonell	tilordne lok	flervareflyt	branch-first, cut-second, D-W	ja
Nõu et al. [1997]	taktisk	tilordne lok	flervareflyt	D-W	ja
Löbel [1997b]	takt./oper.	MDVSP	flervareflyt	LP kolonnegenerering, Lagrangerelaksering, CPLEX, Branch & Cut, heuristikk	nei
Löbel [1997a]	takt./oper.	MDVSP	flervareflyt	D-W	nei
Ziarati et al. [1997]	operasjonell	tilordne lok	flervareflyt	D-W	ja
Kokott and Löbel [1996]	taktisk	MDVSP	flervareflyt	Lagrangerelaksering	nei
Forbes et al. [1994]	taktisk	MDVSP	flervareflyt[b]	LP-relaks., B & B	nei
Schrijver [1993]	operasjonell	tilordne vogner	en/flervareflyt	CPLEX, mincost circulation, konvekse-hull algoritmer	nei
Forbes et al. [1991]	taktisk	en loktype	tilordning	LP-relaks., B & B	nei
Florian et al. [1976]	strategisk	tilordne lok	flervareflyt	Bender	nei

a Vedlikeholdskostnader er korrelert med kilometerkostnader.

b Formulering tilsvarende flervareflyt.

c Operational car assignment problem.

d To modeller, en ordnet flervareflyt og en som ikke er ordnet.

Eksempler på tidsavhengig vedlikehold er Cordeau et al. [2001b,a], Lingaya et al. [2002] og Ziarati et al. [1997].

Problemene som Cordeau et al. [2000, 2001a,b] ser på er et samtidig materielltilordningsproblem der både lokomotiv og vogner skal tilordnes til en ruteplan samtidig.

Cordeau et al. [2000] presenterer en basismodell som er et basert på et flervareflytnettverk. Denne modellen består av et envareflytnettverk for hver materielltype, med felles restriksjoner som binder disse sammen til et flervareflytnettverk. Cordeau et al. [2001b] utvider denne modellen til å inkludere vedlikeholdsrestriksjoner, og sier at dette vedlikeholdskravet kan uttrykkes i form av et maksimalt antall tillatte dager mellom to etterfølgende stopp på en vedlikeholdsbase.

Alle variablene i dette problemet er binære eller heltall. I Cordeau et al. [2000, 2001b] blir problemet forsøkt løst med Benders dekomponeringsalgoritme. For å få algoritmen til å konvergere raskere innføres:

- Et kutt til masterproblemet per subproblem. Subproblemet kan løses for hver materielltype, og kan sende flere kutt til masterproblemet per iterasjon.
- LP-relaksere masterproblemet. Først løse det relakserte masterproblemet. Kutt som blir generert fra det relakserte problemet gjelder også for heltallsproblemet.
- Generere kutt når vedlikeholdskrav er relaksert. Disse kuttene vil være lovlige i det originale problemet også. Fordelen med å gjøre det på denne måten er at kutt blir generert når problemet er enklere å løse.
- Identifisere Pareto-optimale kutt. De ser på de optimale kuttene som blir generert fra ekstremalpunktene, og løser et eget problem for å identifisere Pareto-optimale kutt.
- Løse subproblemet med Dantzig-Wolfe. Subproblemet kan løses med Dantzig-Wolfe når antallet materielltyper øker.

Også Cordeau et al. [2001a] nevner vedlikeholdskravet gitt som at hver sykel må inneholde et periodisk stopp på vedlikeholdsbasen. I tillegg innfører Cordeau et al. [2001a] en restriksjon som passer på at vedlikeholdskapasiteten ved vedlikeholdsbasen ikke blir overskredet. Dette problemet blir løst med Branch & bound og den lineære relakseringen er løst med kolonnegenerering.

3.1.2 Infrastruktur og togrekkefølge på stasjoner

Train unit shunting[1] problemet (*TUSP*) ser på infrastrukturen på stasjonen når tog blir skjøtet, delt og hensatt. Siden et av målene til NSB er riktig rekkefølge på stasjoner, vil TUSP bli kommentert her.

Gjennom natta er målet med TUSP å parkere togsettene på en slik måte at operasjonene på morgenen kan starte opp så smidig som mulig, mens restriksjoner knyttet til parkering, ruter, vask, vedlikehold og personellplanlegging blir møtt [Freling et al.,

[1] shunt = skjøte/dele

2002, Lentink et al., 2003]. I følge Freling et al. [2002] er det lite litteratur på TUSP i dag.

De kompliserende faktorene til TUSP er infrastrukturen på skinnene [Lentink et al., 2003]. Sporene på en stasjon er karakterisert som rekkefølgeavhengige (LIFO) eller frie spor.

For å kunne se på TUSP-problemet, må man se på rekkefølgen av enhetene i toget. Alfieri et al. [2003] ser på rekkefølgen av enheter i hvert tog i et materiellturneringsproblem. Alfieri et al. [2003] ser ikke eksplisitt på vedlikehold, men har vedlikeholdskostnader som er korrelert med antall togenheter og togkilometer. Videre formulerer de problemet som et ordnet flervareflytproblem, i tillegg til en flervareflytmodell som ikke ser på rekkefølgene i togene. De har sett på problemet på en enkelt linje på en enkelt dag. I inndatasettet er det 3000 togserier. Dette problemet har de løst ved å lage en løsningsalgoritme basert på hierarkisk dekomponering, og ved å generere lovlige ulikheter, såkalte valid inequalities, for noen av restriksjonene slik at det blir kutt (restriksjoner) nærmere konvekse hull.

Problemet blir mye større når man tar hensyn til rekkefølgen. I datasettet som Alfieri et al. [2003] ser på er det økning i variabler med en faktor ti, mens antallet restriksjoner øker med en faktor 3,5 i forhold til når rekkefølgen på vognene blir ignorert.

3.2 Lagrangerelaksering

Relakseringer av et problem brukes til å finne grenser for den optimale verdien for problemet. Noen ganger kan relakseringer også bevise optimalitet [Wolsey, 1998].

I et ILP (*Integer Linear Program*) eller et MILP (*Mixed-Integer Linear Program*) kan en eller flere av heltallsvariablene bli relaksert til kontinuerlige variable, mens resten av restriksjonene beholdes i sin opprinnelige form. Dette kalles lineær relaksering.

Ved Lagrangerelaksering relakseres en eller flere av de lineære restriksjonene i stedet for heltallskravet. Dette blir gjort ved at restriksjonene blir vektet inn i målfunksjonen med passende Lagrangemultiplikatorer.

Når man relakserer restriksjoner, vil problemet få et større mulighetsområde, siden restriksjoner blir droppet/fjernet fra problemet. Et større mulighetsområde vil føre til at løsningen vil bli minst like god som den optimale løsningen til det opprinnelige problemet, det vil si en nedre grense for et minimeringsproblem og en øvre grense for et maksimeringsproblem. Kvaliteten på denne nedre (øvre) grensen avhenger av hvilke verdier man velger på Lagrangemultiplikatoren, samt hvilke restriksjoner man velger å relaksere. *Multiplier adjustment* og *subgradient optimization* er to teknikker for å velge verdier på multiplikatoren. Senere i kapitlet kommer mer forklaring på subgradient optimization.

Man har et valg på hvilke restriksjoner man kan velge å relaksere. Et punkt man kan se på er heltallsegenskap[2]. Hvis det Lagrangerelakserte problemet har heltallsegenskap, vil man ikke oppnå en bedre grense enn med lineærrelaksering [Beasley, 1993]. Heltallsegenskap er derfor en attributt å se på for å velge relakseringer. En annen ting man

[2]Heltallsegenskap = Heltallsløsning på variablene når man lineærrelakserer binærkravet til variablene.

kan se på er problemstrukturen. Det kan hende at ved å fjerne noen restriksjoner fra problemet får et problem med en kjent struktur eller enklere struktur.

Standard oppsett for Lagrangerelaksering [Beasley, 1993]:

Starter med den opprinnelige modellen:

$$\begin{aligned} min \quad & cx \\ & Ax \geq b \\ & Bx \geq d \\ & x \in (0,1) \end{aligned} \tag{3.1}$$

Relakserer restriksjonssettet $Ax \geq b$ ved å introdusere en Lagrangemultiplikator $\lambda \geq 0$ som blir lagt til restriksjonssettet og lagt til i målfunksjonen.

Den Lagrange relakserte modellen blir dermed:

$$\begin{aligned} min \quad & cx + \lambda(b - Ax) \\ & Bx \geq d \\ & x \in (0,1) \end{aligned} \tag{3.2}$$

Det lovlige området for (3.2) er minst like stort som mulighetsområdet for (3.1). Når $\lambda \geq 0$ og $b - Ax \leq 0$ vil dette gi et ikke-positivt bidrag til målfunksjonen. Derfor vil modell (3.2) gi en nedre grense på (3.1). Dette problemet kalles *Lagrangean lower bound program* (LLBP).

Det man dermed ønsker her er å finne verdier for multiplikatorene for å få en best mulig nedre grense, det vil si en verdi som er så nær som mulig den optimale løsningen for det opprinnelige problemet.

Denne modellen kalles *Lagrangean dual program*:

$$max_{\lambda \geq 0} \left\{ \begin{array}{c} min \quad cx + \lambda(b - Ax) \\ s.t \quad Bx \geq d \\ x \in (0,1) \end{array} \right\} \tag{3.3}$$

Problemet er å finne en lovlig løsning til det opprinnelige problemet når man har løsingen for det relakserte problemet.

Ved å løse det relakserte problemet kan man i noen tilfeller si noe om den optimale løsningen til det opprinnelige problemet. Hvis det relakserte problemet ikke har løsning, vil ikke det opprinnelige problemet ha noen løsning heller. I andre tilfeller kan man få en løsning av det relakserte problemet som også er optimal for det opprinnelige problemet. Men man kan ikke si at en løsning for LLBP er optimal for det originale problemet selv om den kan være lovlig for det optimale problemet. Løsningen fra LLBP er kun lovlig for det originale problemet hvis og bare hvis [Beasley, 1993]

1. X er lovlig for det originale problemet, og
2. $cX = [cX + \lambda(b - AX)]$, det vil si at $\lambda(b - AX) = 0$

En ulempe ved å bruke et sett av dualvariable til det tilhørende Lagrangeproblemet framfor for eksempel å løse dualproblemet til det opprinnelige problemet er at man må ha en optimal løsning på det relakserte problemet for å kunne si noe om grensen på det opprinnelige problemet. Ved å bruke dualproblemet trenger man kun en lovlig løsning for å kunne si noe om grensen for det opprinnelige problemet [Wolsey, 1998].

3.2.1 Subgradient Optimization

Subgradient optimization er en iterativ prosedyre som fra et startsett med Lagrangemultiplikatorer genererer Lagrangemultiplikatorer videre på en systematisk måte [Beasley, 1993]. Den kan bli sett på som en prosedyre for å forsøke å maksimere den nedre grenseverdien som ble oppnådd i LLBP ved å velge passende verdier på Lagrangemultiplikatorene.

Prosedyren er [Beasley, 1993]:

1. La π være en brukerdefinert parameter som tilfredsstiller $0 < \pi \leq 2$. Initialiser Z_{UB} (for eksempel fra en heuristikk av problemet). Z_{UB} er den beste lovlige løsningen funnet til det originale problemet. Bestem et startsett med Lagrangemultiplikatorer.

2. Løs LLBP med settet av multiplikatorer, for å få en løsning av verdien Z_{LB}.

3. Definer subgradienter G_i for de relakserte restriksjonene, evaluert på den aktuelle løsningen, $G_i = b_i - \sum_{j=1}^{n} a_{ij}X_j, i = 1, ..., m$.

4. Definer en stegstørrelse T ved $T = \frac{\pi(Z_{UB}-Z_{LB})}{\sum_{i=1}^{m} G_i^2}$. Denne stegstørrelsen er avhengig av gapet mellom den nedre grensen Z_{LB} og den øvre grensen Z_{UB} og den brukerdefinerte parameteren π med nevneren som en skaleringsfaktor.

5. Oppdater Lagrangemultiplikatorene, λ_i, ved å bruke $\lambda_i = max(0, \lambda_i + TG_i); i = 1, ..., m$ og gå til steg 2 for å løse LLBP med det nye settet med Lagrangemultiplikatorer.

Slik som denne prosedyren er skrevet terminerer den aldri. Det må altså lages en stoppregel i tillegg. I Beasley [1993] baserer denne stoppregelen seg på antall iterasjoner eller på verdien av π, der π-verdien blir redusert mens prosedyren kjører.

Man kan ikke forvente eller observere en kontinuerlig forbedring i den nedre grensen på hver iterasjon. derfor bør man ha en peker,Z_{max} på den beste grensen som man har oppnådd til enhver tid.

Termineringsregler:

1. Terminere hvis $Z_{max} = Z_{UB}$.

2. Sett $\pi = 2$ til å begynne med. Hvis Z_{max} ikke har forbedret seg på et bestemt antall iterasjoner, halver verdien på π.

3. Hvis man ikke har terminert i trinn 1, terminer prosedyren når π er liten, for eksempel når $\pi < 0.005$.

3.3 Benders dekomponeringsalgoritme

Benders dekomponeringsalgoritme[3] kan brukes når man har et blandet heltallsproblem eller når noen variabler forekommer på en lineær måte mens andre på en ulineær måte i formuleringen av problemet, og hvis problemets struktur indikerer en naturlig oppdeling av variablene [Bender, 1962].

I følge Bender [1962] er ideen til prosedyren å dele det gitte problemet P i to underproblemer, der det ene er et programmeringsproblem som kan være for eksempel lineært, ulineært eller heltallig. Det andre er et lineært programmeringsproblem.

Vi har et maksimeringsproblem (Eksempel er hentet fra [Nygreen, 2004, Rasmussen, 2004, Bender, 1962]):

$$\begin{aligned} max \qquad & z = c^T x + f(y) \\ & Ax + g(y) \leq b \\ & x \geq 0 \\ & x \in R^p \\ & y \in Y \end{aligned} \tag{3.4}$$

I dette problemet er x kontinuerlig i en mengde R^p, mens Y er en mengde med heltall. $f(y)$ og $g(y)$ er funksjoner.

For en gitt verdi på y, $\bar{y}$, reduseres det originale problemet (3.4) til et lineært underproblem med kun kontinuerlige beslutningsvariable. Dette problemet kalles det primale subproblemet:

$$\begin{aligned} max \qquad & c^T x + f(\bar{y}) \\ & Ax \leq b - g(\bar{y}) \\ & x \in R^p \\ & x \geq 0 \end{aligned} \tag{3.5}$$

Dualen til (3.5) er også et lineært problem og kalles Benders duale subproblem:

$$\begin{aligned} min \qquad & [b - g(\bar{y})]^T u + f(\bar{y}) \\ & A^T u \geq c \\ & u \geq 0 \\ & u \in R^m \end{aligned} \tag{3.6}$$

Siden (3.5) er et lineært problem, følger det av dualitetsteoremet at den optimale løsningen av (3.6) vil gi optimal løsning av problem (3.5). Sammenhengen mellom løsningen av primal og dualproblemet er vist i tabell 3.2.

Definisjon 3.1 *En vektor $r \in R^n \backslash \{0\}$ er en stråle i en mengde $P \subseteq R^n$ hvis og bare hvis det gjelder for et vilkårlig punkt $x \in P$ at $\{y \in R^n | y = x + \lambda r, \lambda \in R_+\} \subseteq P$. En ekstrem stråle er en stråle som ikke kan skrives som en kombinasjon av to forskjellige stråler [Rasmussen, 2004].*

[3] Benders dekomponeringsalgoritme blir også kalt L-shaped-metoden [Birge and Louveaux, 1997]. I denne oppgaven bruker jeg kun navnet Benders dekomponering på denne algoritmen.

Tabell 3.2: Sammenheng mellom primalløsning og mulige dualløsninger [Rardin, 1998]

Primal	**Dual**
Optimal	Optimal
Ulovlig	Ulovlig eller ubundet
Ubundet	Ulovlig

La $\mathbf{D}$ være det lovlige området til det duale subproblemet (3.6), og la $P_{\mathbf{D}}$ og $Q_{\mathbf{D}}$være settene av ekstremalpunkter og ekstreme stråler av $\mathbf{D}$.

Hvis det finnes en dual stråle u hvor $[b - g(\bar{y})]^T u < 0$, vil (3.6) være ubundet, (3.5) være uløselig, og dermed vil det originale problemet (3.4) være uløselig for $y = \bar{y}$. Siden man kun er interessert i $\bar{y}$ slik at (3.4) og (3.5) er løselig, vil man sørge for at man kun velger $\bar{y}$ slik at (3.6) blir begrenset [Rasmussen, 2004]. Dette vil bli lagt til som en restriksjon i problemet.

Definisjon 3.2 *Optimalitetskutt vil være kutt av typen $z \leq f(y) + [b - g(y)]^T u^k$ som sørger for at den nedre grenseverdien konvergerer mot den optimale verdien [Rasmussen, 2004].*

Definisjon 3.3 *Lovlighetskutt er kutt av typen $(b - g(y))^T u^k \geq 0$ som sørger for at man ikke får løsninger der subproblemet er uløselig [Rasmussen, 2004].*

Hvis man omformulerer (3.6) på denne måten:

$$w = min_u [b - g(\bar{y})]^T u + f(\bar{y}) \qquad \forall u \in P_{\mathbf{D}} \tag{3.7}$$

og man kjenner alle hjørnepunktene kan man skrive 3.4 på denne måten:

$$\begin{aligned} max \qquad z = f(y) + min_u [b - g(y)]^T u \quad , \qquad & u \in P_{\mathbf{D}} \\ y \in Y \qquad & \end{aligned} \tag{3.8}$$

Hvis vi nå antar at ikke alle hjørnepunktene er kjent, får vi Benders masterproblem:

$$max \qquad z \tag{3.9}$$

$$z \leq f(y) + [b - g(y)]^T u \quad , \qquad u \in P_{\mathbf{D}} \tag{3.10}$$

$$(b - g(y))^T u \geq 0 \quad , \qquad u \in Q_{\mathbf{D}} \tag{3.11}$$

$$y \in Y \tag{3.12}$$

(3.10) er optimalitetskuttene, og (3.11) er lovlighetskuttene. (3.9)-(3.12) vil ha bedre målfunksjonsverdi enn (3.8) fordi (3.9)-(3.12) har færre restriksjoner og dermed et større mulighetsområde.

En løsning av (3.9)-(3.12) gir z og y der $z \geq z^*$, mens en løsning av (3.7) gir $[b - g(\bar{y})]^T u + f(\bar{y}) \leq z^*$ der z^* er den optimale verdien til problemet.

Man vil finne verdier på begge sider av den optimale verdien til problemet. Når løsningene av (3.9)-(3.12) og (3.7) er like, har man en optimal løsning.

I Benders algoritme kan man velge om man vil bruke det duale eller det primale subproblemet så lenge subproblemet er lineært. Bender [1962] har algoritme som tar utgangspunkt i det duale subproblemet. I praksis velger man det problemet som er enklest å løse.

Algoritme 3.1 *Benders algoritme med primalt subproblem [Cordeau et al., 2000].*

1. *Sett t=1 og mengden med ekstremalpunkt, $P_{\boldsymbol{D}}^1$ og mengden med stråler, $Q_{\boldsymbol{D}}^1$ tom.*
2. *Løs Benders masterproblem (3.9)-(3.12).*
   (a) *Hvis problemet er uløselig, vil også det originale problemet (3.4) være uløselig. Stopp algoritmen.*
   (b) *Ellers, la $\bar{y}^t$ være en optimal løsning med verdien UB^t (som vil være en øvre grense på det originale problemets verdi.)*
3. *Løs Benders primale subproblem (3.5) der $\bar{y}^t$ er inputverdi.*
   (a) *Hvis det primale subproblemet er begrenset/har endelig løsning, la x^t være en primal optimal løsning med verdi LB^t og la u^t være dual optimal løsning gitt som et ekstremalpunkt.*
      i. *Hvis $UB^t = LB^t$ er $(\bar{x}^t, \bar{y}^t)$ en optimal løsning til det originale problemet (3.4). Stopp algoritmen.*
      ii. *Ellers er UB^t den øvre grensen til det originale problemet (3.4). Sett $P_{\boldsymbol{D}}^{t+1} = P_{\boldsymbol{D}}^t \cup \{u^t\}$ for å generere et optimalitetskutt. Sett $Q_{\boldsymbol{D}}^{t+1} = Q_{\boldsymbol{D}}^t$*
   (b) *Hvis det primale subproblemet er uløselig la u^t være en ekstrem stråle slik at $(b - g(\bar{y}^t))^T u^t < 0$. Lag et lovlighetskutt ved å sette $Q_{\boldsymbol{D}}^{t+1} = Q_{\boldsymbol{D}}^t \cup \{u^t\}$. Sett $P_{\boldsymbol{D}}^{t+1} = P_{\boldsymbol{D}}^t$*
4. *Sett $t = t + 1$. Gå til trinn 2.*

Den optimale løsningen er når løsningsverdien på subproblemet og masterproblemet er lik hverandre.

Algoritmen vil alltid terminere. Vi har endelig mange ulikheter som blir lagt til i hver iterasjon. Man må bevise at man ikke tilfører det samme kuttet to ganger for å kunne si at algoritmen alltid vil terminere [Rasmussen, 2004]. For lovlighetskutt i trinn 3.b. blir løsningen u kuttet bort, og man vil da ikke komme til samme løsning igjen. Hvis et optimalitetskutt $z \leq f(\bar{y}) + (b - g(\bar{y}))^T u$ har blitt lagt til, vil man når man kommer fram til samme løsning u gå til trinn 3.a.i. siden

$$UB^t \leq f(y) + (b - g(\bar{y}))^T u = LB^t \tag{3.13}$$

og UB^t er en øvre grenseverdi og LB^t er en nedre grenseverdi noe som fører til at $UB^t = LB^t$.

Benders algoritme krever ikke noe spesielt til hvordan formen på $g(y)$ og $f(y)$ er. I praksis må de ha egenskaper slik at problemene kan løses med eksisterende metoder [Bender, 1962].

3.3.1 Lovlighetskutt

Hvis subproblemet ikke er løsbart, ønsker man å legge til en restriksjon til masterproblemet slik at løsningen til masterproblemet tvinges bort fra løsninger som gjør subproblemene uløselig. Det vil si at lovlighetsområdet for mulige løsninger blir mindre og mindre for hvert kutt som blir generert.

Farkas Lemma er noe av teorien som ligger bak forståelsen av lovlighetskutt. Enten vil $\{x \in R_+^n : Ax \leq b\} \neq \emptyset$, eller så vil det eksistere en $v \in R_+^m$ slik at $vA \geq 0$ og $vb < 0$ [Nemhauser and Wolsey, 1988].

Man ønsker å finne den ekstreme strålen som er lengst fra å oppfylle betingelsen. Problemet med å finne retningen på kuttet [Kall and Wallace, 1994, Aschehoug and Fodstad, 2002]:

$$max \qquad \sigma^T(b - g(\bar{y})) \tag{3.14}$$

$$\sigma^T A \leq 0 \tag{3.15}$$

$$\|\sigma\| \leq 1 \tag{3.16}$$

Dersom målfunksjonen er positiv må det lages et lovlighetskutt. Dualen til problemet kan gis på følgende form [Birge and Louveaux, 1997]:

$$Q(\bar{y}) = Min_{x,v^+} \; Iv^+ \tag{3.17}$$
$$g(\bar{y}) + Ax + Iv^+ \geq b \tag{3.18}$$
$$x \geq 0 \tag{3.19}$$
$$x \in R^p \tag{3.20}$$
$$v^+ \in R_+^{m_2} \tag{3.21}$$

I er identitetsmatrisa. Variablene v^+ sørger for at problemet alltid er løsbart. Disse variablene kalles kunstvariable. $Q(\bar{y}) = 0$ for en gyldig $\bar{y}$. Et slikt problem må løses for hvert subproblem, og hvis målfunksjonsverdien er større enn 0 legges det til et lovlighetskutt til masterproblemet på formen $\sigma_k^T g_k(y) \geq \sigma_k^T b$ [Kall and Wallace, 1994]

Dette problemet er fase 1 problemet til subproblemet, og vil alltid gi en lovlig løsning. Denne metoden vil løse et fase1-problem for å sjekke om subproblemet er lovlig ved å legge til kunstvariabler til restriksjonene og minimere summen av disse variablene. Hvis målfunksjonen blir større enn null, betyr det at noen av kunstvariablene har verdi og subproblemet er dermed ikke løselig, og det genereres da et lovlighetskutt. Subproblemet vil være lovlig når målfunksjonen i fase 1 problemet er lik null.

Ubundet dualt subproblem

Fase1-problemet til subproblemet er en måte å finne ut om subproblemet er bundet. McDaniel and Devine [1977] har i stedet for å løse fase1-problemet til subproblemet, valgt å legge til et ekstra sett med restriksjoner til det duale subproblemet for å finne ut om det er ubundet eller ikke.

$$uE \geq M \tag{3.22}$$

der E er en $m \times 1$ vektor av enere, og M er et stort negativt tall som ikke vil være stram for lovlige løsninger av subproblemet. Hvis dette problemet løses, og $uE \geq M$ er en stram restriksjon kan man konkludere med at det duale subproblemet er ubundet og man må legge til lovlighetskutt til masterproblemet.

3.3.2 Optimalitetskutt

Optimalitetskutt baseres på sterk og svak dualitet. Sterk dualitet vil si at i optimum vil målfunksjonen til det primale subproblemet være lik målfunksjonen til det duale subproblemet. Svak dualitet vil si at målfunksjonen til det primale subproblemet vil være mindre (større) enn målfunksjonsverdien til det duale subproblemet når det primale problemet er et maksimeringsproblem (minimeringsproblem) og det duale er et minimeringsproblem (maksimeringsproblem) [Aschehoug and Fodstad, 2002].

Verdien på målfunksjonen til masterproblemet vil derfor være avhengig av løsningene som blir funnet i subproblemet.

3.3.3 Ulikheter

Optimalitetskutt og lovlighetskutt er ulikheter til Benders mastermodell. I dette avsnittet vil jeg beskrive litt teori for ulikheter. Dette for å få formulert ulikhetene til Benders masterproblem på en slik måte at man får en Benders algoritme som kan konvergere raskere.

Lovlige ulikheter og fasetter (*facets*)

Gitt et polyhedron $P = \{x \in R^n : Ax \leq b\}$, er spørsmålet å finne ut hvilke av ulikhetene $a^i x \leq b_i$ er nødvendige for å beskrive P og hvilke som kan kuttes ut. De ulikhetene som er nødvendige for å beskrive P er de samme uavhengig av hvilke initielle ulikheter man har for å beskrive P.

En ulikhet $\pi x \leq \pi_0 \, (\pi, \pi_0)$ er en lovlig ulikhet for P hvis det tilfredsstiller alle punktene i P. Hvis (π, π_0) er en lovlig ulikhet for P og $F = \{x \in P : \pi x = \pi_0\}$, så blir F kalt en *face* av P, og man kan si at (π, π_0) representerer F. En *face* F av P er en fassett (*eng: facet*) til P hvis dim(F) = dim(P) - 1. Hvis F er en fasett av P vil det eksistere noen ulikheter $a^k x \leq b_k$ for $k \in M^{\leq}$ som representerer F.

[Nemhauser and Wolsey, 1988]

I Benders algoritme må man i verste fall generere like mange lovlighetskutt som det er ekstreme stråler til problemet, og hvis man finner en enkel måte å formulere fasettene til problemet vil man kunne løse problemet med Benders algoritme raskere.

Ekvivalente og dominante kutt

To lovlige ulikheter $\pi x \leq \pi_0$ og $\gamma x \leq \gamma_0$ er ekvivalente hvis $(\gamma, \gamma_0) = \lambda(\pi, \pi_0)$ for noen $\lambda > 0$.

Hvis de ikke er ekvivalente og det eksisterer en $\mu > 0$ slik at $\gamma \geq \mu\pi$ og $\gamma_0 \leq \mu\pi_0$, da vil $\{x \in R_+^n : \gamma x \leq \gamma_0\} \subset \{x \in R_+^n : \pi x \leq \pi_0\}$. I dette tilfellet vil man si at $\gamma x \leq \gamma_0$ dominerer eller er sterkere enn $\pi x \leq \pi_0$, eller at $\pi x \leq \pi_0$ er dominert av eller svakere enn $\gamma x \leq \gamma_0$. En maksimal lovlig ulikhet er en som ikke er dominert av en annen lovlig ulikhet [Nemhauser and Wolsey, 1988].

Et kutt er Pareto-optimalt hvis ingen andre kutt dominerer det [Magnanti and Wong, 1981].

Ved å sjekke for dominante og ekvivalente ulikheter kan man redusere antallet kutt i Benders masterproblem.

Chvátal-Gomory prosedyre

Følgende prosedyre blir vist i Roos [2004] for å generere lovlige ulikheter for et heltallsproblem.

$$X = P \cap Z^n, \qquad P = \{x : Ax \leq b, x \in R_+^n\} \tag{3.23}$$

Antar at A er en $m \times n$ matrise med kolonner $a_1, ..., a_n$. La $u \in R_+^m$ være en tilfeldig ikkenegativ m-vektor.

Trinn 1: Siden $b - Ax \geq 0$ og $u \geq 0$, har man at $u^T(b - Ax) \geq 0$. Følgelig er $u^T Ax \leq u^T b$, som er ekvivalent til $(A^T u)^T x \leq u^T b$. Med andre ord har vi ulikheten

$$\sum_{j=1}^{n} (u^T a_j) x_j \leq u^T b \tag{3.24}$$

Trinn 2: Ved å runde av nedover koeffisientene til x_j på venstre siden får man:

$$\sum_{j=1}^{n} \lfloor u^T a_j \rfloor x_j \leq u^T b \tag{3.25}$$

Trinn 3: Venstre siden er nå heltall, så man kan runde ned høyresiden også, og få:

$$\sum_{j=1}^{n} \lfloor u^T a_j \rfloor x_j \leq \lfloor u^T b \rfloor \tag{3.26}$$

I forbindelse med Benders algoritme vil u være den duale variabelen som blir generert fra subproblemet. Lovlighetskuttene i Benders masterproblem har formen $[b - g(y)]^T u \geq 0$. Siden $u \geq 0$ vil også $[b - g(y)]^T \geq 0$. Da kan man formulere hvert lovlighetskutt med prosedyren som er vist her for å få kuttene nærmere konvekse hull.

Kapittel 4

Vedlikeholdsproblemet

I dette kapitlet presenteres modellen for vedlikeholdsproblemet. For nærmere diskusjon rundt modelleringen henvises leseren til Bjorvand and Risberg [2004]. Deretter vil vedlikeholdsproblemet bli formulert med løsningsalgoritmene fra forrige kapittel før et valg av løsningsalgoritme vil bli foretatt.

4.1 Modellformulering

Modellen blir stor når man betrakter hele materiellturneringsproblemet inkludert kravene til vedlikehold. I Aschehoug and Fodstad [2002] ble materiellplanleggingsproblemet til lokaltogtrafikken på Østlandet modellert og implementert, men vedlikeholdskrav ble utelatt og løsningstiden for modellen var høy. Siden modellen hadde lang løsningstid vil det i første omgang være mest hensiktsmessig å løse hele materiellturneringsproblemet til NSB i to modeller. Den første modellen kan bygge på problemet som blir sett på i Aschehoug and Fodstad [2002] og som tar hensyn til alle krav til materiellturneringsproblemet utenom krav til vedlikehold. Denne modellen lager dagsløp som den andre modellen kan ta som inndata. Den andre modellen vil lage en materiellturneringsplan der vedlikehold er tatt hensyn til, i tillegg til de kravene til materiellturneringsproblemet som ikke blir tatt hensyn til igjennom inndataen. Det er denne andre modellen, heretter kalt vedlikeholdsmodellen eller vedlikeholdsproblemet, som vil bli studert videre i denne oppgaven.

Problemet blir modellert med en nettverksmodell med utgangspunkt i Bjorvand and Risberg [2004]. Fra Bjorvand and Risberg [2004] har jeg valgt å ta utgangspunkt i modellen som kan planlegge for flere uker, Flerukerproblemet, enten like eller ulike uker. Det betyr at man kan planlegge med en periode som er lenger enn en uke, for å se om det blir færre vedlikehold hvis planhorisonten er flere uker i stedet for kun en uke. Da kan man velge om dagsløpene i disse ukene skal kobles i samme rekkefølge påfølgende uker eller ikke. I tillegg er det i dette problemet også tatt hensyn til at man kan ha flere materielltyper.

4.1.1 Inndata

Det vil være to typer noder i modellen, dagsløpsnoder og vedlikeholdsnoder. Hvert mulige vedlikehold på vedlikeholdsbasen modelleres som en egen node. Vedlikeholdsnodene opprettes ved at det tas hensyn til når det er muligheter for vedlikehold på hver enkelt base og hver vedlikeholdsnode er en vedlikeholdsmulighet.

Koblingene mellom nodene sier hvilke noder som skal kobles sammen, det vil si hvilke noder som skal kjøres etter hverandre av samme materiellenhet.

For hver node må man vite:

- Startstasjon
- Endestasjon
- Starttidspunkt
- Sluttidspunkt
- Ukedag
- Hvilke togtyper kan entre noden

I tillegg må man for dagsløpene ha:

- Lengden på dagsløpet i antall kilometer.

For hver materielltype må man vite

- Hvor langt materiellet kan kjøre mellom to vedlikehold (Kilometergrense).
- Ved hvilke vedlikeholdsbaser materiellet kan vedlikeholdes.
- Antallet av hver materielltype (materielltilgjengelighet).

På vedlikeholdsbasene trenger man denne informasjonen for å kunne generere vedlikeholdsnoder:

- Hvilke togtyper som kan vedlikeholdes.
- Tidspunkter for hvert skift.
- Vedlikeholdskapasiteten.

For hver stasjon (en vedlikeholdsbase regnes også som en stasjon) må man vite

- Overnattingskapasiteten for hver ukedag.
- Avstandene til de andre stasjonene.

Inndataen legges til for en uke. Hvis man ønsker å kjøre problemet for flere uker, kopieres informasjonen automatisk. Hver uke vil da inneholde de samme nodene.

4.1.2 Notasjon

Her presenteres en oversikt over alle indekser, konstanter og variabler som blir brukt i modellformuleringen.

Indekser:

i, j	Node
s	Stasjon
t	Dag
k	Materielltype
h	Hovedmaterielltype
u	Uke

Sett:

S	Stasjoner
N	Noder
D	Dagsløpsnoder, $D \subset N$
V	Vedlikeholdsnoder, $V \subset N$
O_j	Lovlige etterfølgernoder (utnoder) for node j, $O_j \subset N$.
I_j	Lovlige forgjengernoder (innoder) for node j, $I_j \subset N$.
T	Dager i en grunnuke
U	Uker i planleggingsperioden
H	Hovedmaterielltyper
K	Materielltyper (hver subtype av materiellet er en egen materielltype)
K_j	Sett av materielltyper som kan entre node j.

Konstanter:

$KMGRENSE^k$	Maksimalt antall tillatte kjørte kilometer mellom to vedlikehold for materielltype k.
$LENGDE_j$	Tabell med lengden av dagsløpsnode $j \in D$ gitt i antall kilometer.
$STARTSTED_j$	Stasjonen der node j starter.
$SLUTTSTED_j$	Stasjonen der node j slutter.
$STARTTID_j$	Starttidspunkt for node j.
$SLUTTID_j$	Sluttidspunkt for node j.

$KMTT_{ij}$ Avstanden i kilometer mellom $SLUTTSTED_i$ og $STARTSTED_j$. For noder der $SLUTTSTED_i = S\ TARSTED_j$ vil $KMTT_{ij}$ være null. Ellers vil denne lengden angi tomtogskjøring i kilometer.

$TIDTT_{ij}$ Avstanden i tid mellom $SLUTTSTED_i$ og $STARTSTED_j$. For noder der $SLUTTSTED_i = STARSTED_j$ vil $TIDTT_{ij}$ være null.Ellers vil denne lengden angi tiden det tar å kjøre denne tomtogskjøringen.

G_j Ukedag for dagsløp j.

T_{MAX} Antall dager i en grunnuke.

$NODETYPE_j^k$ Tabell som sier hvilke(n) materielltype(r) som kan entre noden j.

$HKAP_{st}$ Maksimalt antall sett det er lov til å hensette ved stasjon s på dag t. Antas lik uke for uke.

$MATKAP^k$ Tilgjengelig materiell av type k.

U_{MAX} Antall uker i planhorisonten.

A_{ij} Hvis nodene $i \in D$ og $j \in D$ må kjøres av samme hovedmaterielltype, vil dette markeres i denne tabellen med tallet 1. Hvis de er uavhengige er tallet 0.

C^V Kostnad ved et vedlikehold.

C^{TT} Kostnad ved å kjøre tomtog per kilometer.

Beslutningsvariable:

Flyt:

$x_{iju}^k =$ $\begin{cases} 1 & \text{hvis node } i \text{ kobles til node } j \text{ for materielltype } k \text{ i uke } u \\ 0 & \text{ellers} \end{cases}$

Kilometer:

$z_{ju}^k =$ avstanden siden forrige vedlikehold gitt at dagsløp j er gjennomført i uke u av materielltype k

Tomtog:

$tt_{isju}^k =$ forteller ved hvilken stasjon toget overnatter når node i og j blir koblet sammen

4.1.3 Generering av subsett

For å lage nettverket gjennomføres det en preprosessering der man lager subsett av lovlige noder å koble sammen. Subsettene blir laget for å redusere antall variabler og restriksjoner. Det vil si at det lages subsett av noder til hver enkelt node som inneholder de nodene det er lov å koble sammen med denne noden. For hver node er det laget to subsett, et sett med *innoder* som inneholder noder det er lov å koble inn til den aktuell noden og et sett med *utnoder* som inneholder noder det er lov å koble fra den aktuell noden. Kriteriene for å lage disse subsettene er:

- De to dagsløpene må være på samme dag, eller på to etterfølgende dager.
- Det må være tid nok mellom dagsløpsnodene og vedlikeholdsnodene til at samme sett kan kjøre begge nodene, og eventuelt tid til å kjøre tomtog hvis de stopper og starter på forskjellig stasjon. Hvis i skal være en lovlig innode til node j, må følgende krav oppfylles: $S_{LUTTID_i} + T_{IDTT_{ij}} \leq S_{TARTTID_j}$, der $T_{IDTT_i}j$ er tiden det tar å kjøre et tomtog mellom endestasjon for node i og startstasjon for node j.
- Nodene som kobles sammen må kjøres av samme materielltype.
- Vedlikeholdsnode kan ikke være innode eller utnode til en annen vedlikeholdsnode. At et vedlikehold ikke kan kobles til et annet vedlikehold er utelatt i denne prosesseringen for å redusere antallet variabler og restriksjoner. I en optimal løsning vil modellen ikke koble et vedlikehold direkte til et vedlikehold uansett, siden det da blir et vedlikehold ekstra i modellen.

4.1.4 Antakelser

I dette avsnittet vil jeg forklare hvilke antakelser og forenklinger av virkeligheten som er gjort i modellen.

Driftspausebasert vedlikehold. Det blir sett bort i fra tungt vedlikehold. Ved tungt vedlikehold tas materiellet ut av drift og erstattes med et annet materiellindivid. Det blir også sett bort i fra korrektivt vedlikehold. Modellen begrenses til å se på vedlikehold som tas i driftspauser i materiellturneringen, driftspausebasert vedlikehold. Det forutsettes at materiellreserven er tilstrekkelig slik at man kan erstatte materiellet som skal ut til tungt vedlikehold og at korrektivt vedlikehold kan tas i driftspauser der det ikke er planlagt vedlikehold.

Vedlikeholdsgjennomføring På vedlikeholdsbasene inndeles mulige vedlikehold innenfor et skift i firetimers bolker, et skift er på åtte timer. Modellen tar ikke hensyn til at man kan dele opp et vedlikehold. Det betyr at man vil fullføre et vedlikehold før man begynner på det neste vedlikeholdet.

Kostnader i modellen. Har satt at kostnadene for hvert vedlikehold er større enn summen av alle kostnadene for tomtogskjøring. Dette fordi hovedmålet til modellen er å minimere antall vedlikehold. Kostnadene gjenspeiler derfor ikke de reelle kostnadene.

Tomme dagsløp. Dette er dagsløp som ikke har ruteplanlagte turer i dagsløpet. Det betyr at togsettet står rolig på en stasjon hele dagen. Disse dagsløpene er gitt med en stasjon som de står på. Jeg har valgt å beholde stasjonen fra NSBs plan, siden hensettingskapasiteten er vanskelig å få inn i modellen, se diskusjon senere i kapitlet. For disse dagsløpene har jeg sagt at alle vedlikehold på samme dag er lovlige inn og utnoder for dagsløpet. Disse nodene vil være i både inndatasettet og utdatasettet til en vedlikeholdsnode på samme dag.

4.1.5 Restriksjoner

Her presenterer restriksjonene til vedlikeholdsproblemet.

Nettverk

Restriksjonene for nettverket er:

$$\sum_{i \in I_j} x^k_{iju} - \sum_{i \in O_j} x^k_{jiu} = 0 \qquad \forall j \in N, k \in K_j, u \in U \tag{4.1}$$

$$\sum_{k \in K_j} \sum_{i \in I_j} x^k_{iju} = 1 \qquad \forall j \in D, u \in U \tag{4.2}$$

$$\sum_{k \in K_j} \sum_{i \in I_j} x^k_{iju} \leq 1 \qquad \forall j \in V, u \in U \tag{4.3}$$

Restriksjon (4.1) sier at flyten inn i en node skal være lik flyten ut av noden og denne restriksjonen sørger for at det dannes sykler i modellen. Alle dagsløpene skal dekkes, og det sørger restriksjon (4.2) for. Restriksjon (4.3) sier at hver vedlikeholdsnode kan brukes høyst en gang. Nettverksrestriksjonen for vedlikeholdsnodene passer også på at vedlikeholdskapasiteten på vedlikeholdsbasen ikke blir overskredet, siden hvert mulige vedlikehold modelleres som en egen node.

Kilometergrense

For å sørge for at togsettene kjøres inn til vedlikehold før de overstiger kilometergrensen, må det legges til en kilometerteller i modellen. Denne teller opp alle kilometerne materiellet har kjørt siden forrige vedlikehold, både tomtogskilometer og dagsløpskilometer. Det vil være litt forskjellige tellerrestriksjoner ettersom hvilke to noder man kobler sammen.

$$x^k_{iju}(z^k_{iu} + KMTT_{ij} + LENGDE_j - z^k_{ju}) \leq 0 \qquad \forall j \in D, i \in D \cap I_j, k \in K_j, u \in U \tag{4.4}$$

$$x^k_{iju}(z^k_{iu} + KMTT_{ij} - KMGRENSE^k) \leq 0 \qquad \forall j \in V, i \in I_j, k \in K_j, u \in U \tag{4.5}$$

$$x^k_{iju}(KMTT_{ij} + LENGDE_j - z^k_{ju}) \leq 0 \qquad \forall j \in D, i \in V \cap I_j, k \in K_j, u \in U \tag{4.6}$$

$$0 \leq z_{ju}^{k} \leq KMGRENSE^{k} \qquad \forall j \in D, k \in K_j, u \in U \tag{4.7}$$

$$x_{iju}^{k} \in \{0,1\} \qquad \forall j \in D, i \in I_j, k \in K_j, u \in U \tag{4.8}$$

Restriksjon (4.4) sier at hvis dagsløp i og dagsløp j kobles sammen, så må telleren etter å ha kjørt dagsløp j være minst lik teller etter dagsløp i pluss lengden til dagsløp j og eventuell tomtogskjøring mellom dagsløpene. Restriksjon (4.5) sier at når man kjører inn til vedlikehold, så må telleren pluss kjøringen til vedlikehold være mindre enn kilometergrensen. Når man kjører fra vedlikehold vil telleren være null, og derfor vil restriksjon (4.6) si at telleren etter å ha kjørt første dagsløp må være tomtogskilometer fra vedlikehold, samt lengden av dagsløp j. Restriksjonene (4.7) og (4.8) er krav til variablene.

Kilometertellerrestriksjonene (4.4)-(4.6) er ulineære, og må omformuleres til lineære restriksjoner hvis programvare som for eksempel XpressMP skal kunne løse problemet. Man bruker BIG M-variabler og omformulerer disse restriksjonene slik at restriksjonene blir så stramme som mulig. Diskusjon rundt omformuleringen av disse kravene finnes i Bjorvand and Risberg [2004]. Her er kun omformuleringen gjengitt.

$$z_{iu}^{k} - z_{ju}^{k} + M1_{ij}x_{iju}^{k} \leq M1_{ij} - KMTT_{ij} - LENGDE_j \qquad \forall j \in D, i \in D \cap I_j, k \in K_j, u \in U \tag{4.9}$$

$$z_{iu}^{k} + M2_{ij}x_{iju}^{k} \leq M2_{ij} - KMTT_{ij} + KMGRENSE^{k} \qquad \forall j \in V, i \in I_j, k \in K_j, u \in U \tag{4.10}$$

$$M3_{ij}x_{iju}^{k} - z_{ju}^{k} \leq M3_{ij} - KMTT_{ij} - LENGDE_j \qquad \forall j \in D, i \in V \cap I_j, k \in K_j, u \in U \tag{4.11}$$

BIGM-konstantene vil ha verdiene:

$$\begin{aligned} M1_{ij}^{k} &= max\{z_i + KMTT_{ij} + LENGDE_j - z_j\} \\ &= KMGRENSE^{k} + KMTT_{ij} + LENGDE_j \end{aligned} \tag{4.12}$$

$$M2_{ij} = max\{z_i + KMTT_{ij} - KMGRENSE^{k}\} = KMTT_{ij} \tag{4.13}$$

$$\begin{aligned} M3_{ij} &= max\{KMTT_{ij} + LENDGE_j - z_j\} \\ &= KMTT_{ij} + LENGDE_j \end{aligned} \tag{4.14}$$

Når det er flervareflyt i nettverk er man ikke garantert en heltallsløsning [Rardin, 1998]. Samtidig som det er flervareflyt i nettverk vil restriksjonene (4.9)-(4.11) føre til at modellen ikke har heltallsegenskap, og restriksjon (4.8) kan dermed ikke lineærrelakseres. For nærmere diskusjon av dette henvises leseren igjen til Bjorvand and Risberg [2004].

Materiellt tilgjengelighet

I utgangspunktet blir dette kravet tatt hensyn til når man lager dagsløpene. Men fordi dagsløpene i inndataene kan være ufullstendige (deler av dagsløp) må dette kravet også sjekkes i vedlikeholdsmodellen. Dette kravet kan man sørge for at oppfylles når man teller alle koblingene på et bestemt sted i nettverket, for eksempel ved et døgnskille, og sier at dette antallet må være lavere enn eller lik materiellt tilgjengeligheten. På grunn av syklisk egenskap til modellen vil da materiellt tilgjengeligheten være ivaretatt i hele planen.

$$\sum_{j \in N | G_j = 1} \sum_{i \in I_j | G_i = T_{MAX}} x^k_{iju} \leq MATKAP^k \qquad \forall k \in K, u \in U \tag{4.15}$$

Restriksjon (4.15) teller alle koblingene over døgnskillet mellom siste og første dag i sykelen, og det kan like gjerne være et annet døgnskille. Antallet kan ikke overstige materiellt tilgjengeligheten. Materiellt tilgjengelighet er gjerne gitt som antall materiellenheter som skal til for å kjøre materiellturneringsplanen når ikke vedlikehold er tatt hensyn til fordi man ønsker å bruke minst mulig materiell.

Toglengde

Dette kravet er oppfylt i inndataene, fordi dette blir tatt hensyn til når man bestemmer hvilke togtyper som kan kjøre hver tur og setekapasiteten per tur, og hvilke sett som kjører sammen. Det vil derfor ikke være behov for å modellere denne restriksjonen når dagsløp er inndata. Heretter vil denne restriksjonen bli sett bort i fra.

Snuing, skjøting og deling

Ut i fra inndataene er det bestemt hvor snuinger, skjøtinger og delinger skal skje inni dagsløpene. Hvis det er behov for snuing eller skjøting og deling på starten eller slutten av dagsløpene vil ikke modellen ta hensyn til dette. Dette kan kun inkluderes i modellen ved at man inkluderer en "minstetid" mellom to noder når man lager subsettene av lovlige noder å koble sammen. Dagsløpene i seg selv gir ikke noe informasjon om når det er behov for tid til snuing eller skjøting og deling.

Løsningen man får ved å inkludere en "minstetid" vil alltid være lovlig for det virkelige problemet. Men man kan være uheldig å utelukke noen koblinger som vil være lovlig i det virkelige problemet, og løsningen kan derfor bli dårligere. I modellen som er implementert har jeg ikke tatt hensyn til en "minstetid", men det er enkelt å inkludere.

Hensettingskapasitet og overnattingskapasitet

Denne restriksjonen er tatt hensyn til på dagen gjennom inndataen hvis man tar inn hele dagsløp. Siden modellen også tar inn deler av dagsløp, trenger ikke denne restriksjonen lenger å være oppfylt. På grunn av dagsløpenes egenskaper vet man ikke hvor hvert enkelt togsett er til enhver tid, og hensettingskapasiteten vil være umulig å modellere på dagtid for vedlikeholdsproblemet. Overnattingskapasitet (hensettingskapasitet på natt) vil kunne modelleres, siden alle dagsløpene slutter på nattestid. Denne restriksjonen

er ikke tatt hensyn til gjennom inndataene siden tomtogskjøringen fra de opprinnelige dagsløpene er fjernet.

For å få en restriksjon på dette kravet, må tomtogskjøring gjøres mer fleksibel enn det som ble gjort med tellerrestriksjonene og i Bjorvand and Risberg [2004]. Problemet der er at enten står toget på sluttstedet til det ene dagsløpet eller startstedet på det neste. Men det er noen stasjoner som ikke kan ha overnatting, og hva skjer hvis overnattingskapasiteten blir overskredet? Man må derfor ha mulighet for at togene kan overnatte på en tredje stasjon, som verken er sluttsted eller startsted for nodene. Tomtogsberegningene kan derfor skrives som $K_{MTT_{is}}+K_{MTT_{sj}}$ i stedet for $K_{MTT_{ij}}$ der s er stasjonen som togsettet overnatter på, og i og j er nodene som skal kobles sammen.

I tillegg trenger vi variabler som sier noe om når tomtogene kjøres, og eventuelt ved hvilken stasjon de overnatter.

Man kan derfor innføre følgende restriksjon:

$$x^k_{iju} = \sum_{s \in S} tt^k_{isju} \qquad \forall j \in N, i \in I_j, k \in K_j, u \in U \tag{4.16}$$

Restriksjon (4.16) sier at for hver flytvariabel som kobler sammen to noder også opprettes en variabel med en indeks s i tillegg til i, j, u for å fortelle ved hvilken stasjon toget hensettes. Overnattingskapasitetsrestriksjonen vil da være:

$$\sum_{j \in N} \sum_{i \in I_j | G_j \neq G_i} \sum_{k \in K_j} tt^k_{isju} \leq H_{KAP_{st}} \qquad \forall t \in T, u \in U, s \in S \tag{4.17}$$

$$tt^k_{isju} \in \{0, 1\} \qquad \forall j \in N, i \in I_j, k \in K_j, s \in S, u \in U \tag{4.18}$$

Restriksjon (4.17) summerer alle togene som står på en stasjon over natta. Denne restriksjonen gjelder da bare for noder som er på forskjellige dager. Slik som denne restriksjonen nå er satt opp, er hensettingskapasiteten uavhengig av hvilke materielltyper som blir hensatt. I virkeligheten er lengden på settene forskjellig, og det kan føre til at kapasiteten ikke er uavhengig av materielltype. I tillegg må restriksjonene (4.9)-(4.11) endres med hensyn på $K_{MTT_{ij}}$-tabellen hvis hensettingskapasitet skal inkluderes.

Ved implementasjon av fleksibelt tomtog og hensettingskapasitet blir modellen større med hensyn på restriksjoner og variabler. Det blir innført variabler mellom hver node til hver stasjon og fra hver stasjon til hver node igjen. Samtidig blir det en økning i antall restriksjoner, en faktor tilsvarende antall stasjoner. I vedlegg E har jeg lagt ved noen resultater fra testingen med hensettingskapasitet, og problemet er formulert og løst i XpressMP.

Man må foreta en avveining mellom løsningstid og modellkvalitet når det gjelder kravet til hensettingskapasitet. Hvis tomtogskjøringen gjøres fleksibel, så kan allikevel en materiellplanlegger finne en bedre løsning enn modellen vil klare, fordi han kan se på muligheten for å kjøre flere tomtog sammen og muligheten for å kjøre tomme tog sammen med tog som kjører ruter. Hvis man først har bestemt at det skal kjøre et tomtog mellom to stasjoner, vil de fleste kostnadene være relatert til personellet. En ordentlig kostnadsoversikt eksisterer ikke i dag i NSB, og hvordan man velger å forflytte materiellet vil være avhengig av hvordan personellturnusene er. Dette vil være vanskelig å inkludere i en modell. Når i tillegg hensettingskapasitet øker løsningstiden, se vedlegg E, har jeg

derfor valgt i samråd med NSB å utelukke hensettingskapasiteten fra modelleringen i første omgang. Tomtogskjøringen vil derfor bare bli identifisert, og materiellplanleggeren må i ettertid sjekke gyldige løsninger.

Sportilgang

Dette er et krav som ikke NSB selv har full kontroll over. Det er Jernbaneverket som tar avgjørelser når det gjelder hvilke tog som får kjøre på hvilke jernbanespor til gitte tidspunkt. NSB setter opp et forslag til rutetider og materiellbruk, sender til Jernbaneverket, og får tilbake en justert rutetabell. Når modellen har foreslått tomtogskjøring, og materiellplanleggeren har funnet ut når det er best å kjøre dette tomtoget må det klareres med Jernbaneverket om det er mulig og bestilles rute.

Etterspørsel

Etterspørselen er dekket av inndataene til vedlikeholdsproblemet, da det er bestemt hvilke materielltyper som kan kjøre de ulike turene. For å finne hvilken materielltype som kan kjøre de ulike turene sjekkes setekapasitet og andre egenskaper ved materielltypen som hastighet og akselerasjon. Hvert dagsløp kan kun dekkes av et togsett. Denne restriksjonen blir derfor overflødig i vedlikeholdsproblemet.

Kompatible typer

Hvis flere materielltyper tilfredsstiller kravene til å kjøre et dagsløp, må det inkluderes krav til kompatible typer.

Så lenge dagsløpsnodene er gitt ved kun en bestemt materielltype, vil dette kravet allerede være tatt hensyn til. Hvis ikke alle dagsløpene er gitt med materielltype, men man kan velge mellom flere typer, må kravet til kompatibilitet inkluderes. Dette fordi det ofte er mer enn et togsett som kjører sammen på en tur, og disse togsettene som kjører samme tur må være kompatible. Det kan være ulike subtyper som kjører sammen, men det må være samme hovedmateriellype.

Som inndata til modellen må man for hvert dagsløp også vite hvilke andre dagsløp det turnerer sammen med, det vil si hvilke dagsløp det er avhengigheter mellom.

$$\sum_{k \in K \cap h} \sum_{l \in I_i} x^k_{ilu} - \sum_{k \in K \cap h} \sum_{l \in I_j} x^k_{jlu} = 0 \qquad \forall (i,j) \in D \times D | A_{ij} = 1, h \in H, u \in U \quad (4.19)$$

Restriksjon (4.19) sier at hvis dagsløp inneholder turer der flere togsett kjører sammen, er de avhengig av å kjøres med kompatible typer. A_{ij} er tabell og lik en hvis i og j kjører minst en tur sammen, og det betyr at i og j må kjøres av samme hovedmateriellype. H er et sett med hovedmateriellypene (uten subtyper), og ved å summere alle subtypene per hovedmateriellype får man inkludert kravet om kompatible typer.

Dette kravet er enkelt å implementere i modellen. Jeg har allikevel valgt å se bort fra dette, siden det er planlagt at man skal løse vedlikeholdsproblemet når man har materiellet gitt, og da vil man allerede ha valgt hvilket materiell som skal kjøre de ulike turene. I tillegg har man et mål om at det skal være forutsigbart for kunden hva

slags materielltype som kjører de ulike strekningene, derfor vil man stort sett kun ha en hovedmaterielltype gitt per strekning.

I tillegg må A_{ij}-tabellen genereres manuelt ved å studere hvert enkelt dagsløp nærmere.

Dette kravet kan være aktuelt å ta med hvis man for eksempel skal generere flere planer for å utføre en konsekvensanalyse, og man ønsker å se på det totale antallet vedlikehold hvis man bytter litt om på hvilke typer som kan kjøre de ulike dagsløpene.

Infrastruktur på stasjonene

I NSB har det blitt mer fokus på hvordan infrastrukturen er på stasjonen i den senere tid når man skal planlegge en materiellturnering. Jeg har ikke funnet noen artikkel som ser på både rekkefølgen på stasjonene og vedlikehold, siden problemet blir komplekst med disse modellutvidelsene.

Ved å se på infrastruktur mener jeg at man ser på rekkefølgen av tog som ankommer og forlater en stasjon. Avhengig av hvordan sporene er på stasjonen, kan de karakteriserer som sist inn - først ut, først inn - først ut eller fri. Ved å ta hensyn til infrastrukturen vil operasjonene på hver stasjon gå smidigere.

En forutsetning for å kunne se nærmere på dette kravet er at man vet rekkefølgen av tog inn og ut på hver stasjon. Slik som tomtogskjøringen er modellert vil ikke dette være tilfellet. Samtidig vet man ikke på bakgrunn av egenskapene til dagsløpet hvem som kommer først hvis det er flere tog som ankommer en stasjon på samme minutt, eller som kjører sammen.

Jeg velger derfor å se bort i fra dette kravet siden vedlikeholdskrav er hovedfokus i denne oppgaven, og jeg ikke har funnet noen artikler i litteraturstudiet mitt som klarer å løse dette kravet enkelt. Når man har løst vedlikeholdsproblemet kan man begynne å se på dette kravet for at planen skal bli bedre tilpasset virkeligheten og oppfylle målene til materiellturneringsplanen.

Like uker

Dette kravet betyr at hvis man planlegger for flere uker, kan man ønske at de samme dagsløpene skal kobles sammen i de ulike ukene. Restriksjon (4.20) sørger for at koblingene mellom dagsløp er like etterfølgende uker, men de kan kobles med et vedlikehold mellom.

$$\left(x^k_{iju} + \frac{1}{2}\sum_{l \in V^k \cap O_i \cap I_j} (x^k_{ilu} + x^k_{lju})\right) - \left(x^k_{ij(u-1)} + \frac{1}{2}\sum_{l \in V^k \cap O_i \cap I_j} (x^k_{il(u-1)} + x^k_{lj(u-1)})\right) = 0$$
$$\forall j \in D, i \in D \cap I_j, k \in K_j, u \in (2, ..., U_{MAX}) \quad (4.20)$$

4.1.6 Målfunksjon

Rene pendler kan man tenke på når man planlegger dagsløpene, men det er vanskelig å få dette inn i denne modellen siden vi inne i hvert dagsløp ikke vet hvilke turer og hvilke strekninger det inneholder. Hvis dette skal tas hensyn til, må hvert dagsløp ha en

indeks som sier hvilken pendel turene i dagsløpet kjører, og settes en kostnad på å koble sammen dagsløp på forskjellige pendler. Jeg velger å se bort i fra denne målsettingen i første omgang.

Syklisk turnering blir det automatisk når det blir laget en grunnuke og vi har flytkonservering på hver enkelt node. Da vil det bli en grunnuke som vil gjentas uke for uke. Men antallet uker i hver sykel er det ingen restriksjon på.

At dagsløpsnummer skal følge etter hverandre gjennom uken er ikke tatt hensyn til.

Antall enheter av materiell som blir brukt vil ikke minimeres i denne modellen, men det er mulig å inkludere det. I stedet settes det opp en restriksjon på materiellTilgjengelighet, siden man vet etter man har løst materiellturneringsproblemet hvor mange materiellenheter man trenger og dette antallet blir materiellTilgjengeligheten i vedlikeholdsproblemet. Det går an å legge til en kostnad på hvert materiell, og legge til i målfunksjonen hvis man ønsker det.

Bidragene til målfunksjonen i problemet jeg skal modellere her er tomtogskjøring og antall vedlikehold. Kostnadene ved vedlikehold settes mye større enn kostnadene ved tomtogskjøring. Dette fordi hovedfokuset til modellen er å minimere antall vedlikehold. Gitt et antall vedlikehold ønsker man å minimere tomtogskjøringen og legge kilometertellerne på sine nedre grenser. Målfunksjonen vil da være:

$$min \qquad \sum_{j\in V}\sum_{i\in I_j}\sum_{k\in K_j}\sum_{u\in U} C^V x^k_{iju} + \sum_{j\in N}\sum_{i\in I_j}\sum_{k\in K_j}\sum_{u\in U} C^{TT} K_{MTT_{ij}} x^k_{iju} + \sum_{j\in D}\sum_{k\in K_j}\sum_{u\in U} z^k_{ju} \tag{4.21}$$

C^V settes mye høyere enn C^{TT}. I min implementering har jeg sagt at $C^V = 1$ og $C^{TT} = \frac{1}{100000}$.

Hvis man ønsker å sette ulike kostnader på de ulike vedlikeholdene, kan dette gjøres ved å gjøre om C^V til en tabell med indeks j der j er en vedlikeholdsnode. Tilsvarende kan man sette ulike kostnader på tomtog på ulike strekninger. Jeg ikke prøvd med ulike kostnader i implementering og testing for å se hva dette har å si for løsning og løsningstid.

Det siste leddet i målfunksjonen er tatt med for å sette telleren så lav som mulig. Hvis dette leddet ikke er tatt med, vil modellen bare sørge for at den ligger innenfor $K_{MGRENSE}{}^k$, men ikke noe mer enn det. Hvis materiellplanleggeren ønsker å sjekke telleren mellom dagsløpene for å se hvordan man skal løse tomtogsproblematikken, er det best om denne telleren er riktig, og ikke bare er innenfor grensene, slik at materiellplanleggeren vet den virkelige lengden som er kjørt etter hvert dagsløp.

Men det er et problem med å inkludere telleren i målfunksjonen. Ved å minimere telleren, vil man prøve å gi så lik slitasje til hvert sett som mulig. Dette vil øke løsningstiden i modellen mye, så jeg velger heller å løse et tilleggsproblem. Når man har koblingene og vedlikeholdene gitt, ønsker jeg å minimere hver teller. Dette kan gjøres ved å løse Benders primale subproblem (som jeg kommer tilbake til senere i dette kapitlet) etter at man har bestemt koblingene mellom alle nodene.

Målfunksjonen vil derfor være:

$$min \qquad \sum_{j\in V}\sum_{i\in I_j}\sum_{k\in K_j}\sum_{u\in U} C^V x^k_{iju} + \sum_{j\in N}\sum_{i\in I_j}\sum_{k\in K_j}\sum_{u\in U} C^{TT} K_{MTT_{ij}} x^k_{iju} \tag{4.22}$$

4.2 Modell oppsummering

De kravene som blir inkludert i modellen er:

- Alle vedlikeholdskrav er ivaretatt. Ingen tog kjører mer enn tillatt kilometergrense mellom to vedlikehold. Kapasiteten på vedlikeholdsbasene er overholdt siden hvert mulige vedlikehold blir modellert som en egen node.
- Syklisk løsning siden dagsløpene på siste dag i uken kobles til dagsløp først i uken.
- Materiellbruken overstiger ikke materiellтilgjengeligheten.
- Tomtogskjøring. Det er lov til å kjøre tomtog mellom to noder hvis de ikke ender og starter på samme stasjon. (Har lagt inn dette kravet slik at man kan velge om tomtogskjøring skal være tillatt eller ikke.)
- Vedlikehold på dag kan prioriteres før vedlikehold på kveld og natt. (Gjøres ved å sette kostnad på hver vedlikeholdsmulighet.)

De kravene som ikke blir inkludert i modellen er:

- Hensettingskapasitet[1].
- Sist inn - først ut (LIFO) for togsettene ved de stasjonene der det er et krav.
- Krav til maksimal lengde på sammenhengende sykel.
- Et dagsløpsnummer skal helst følge etter det samme dagsløpsnummeret i hver kobling.
- Kompatibilitet, hvert dagsløp kan kun dekkes av en forhåndsbestemt materiell-type.

Modellen jeg ønsker å se nærmere på i denne diplomoppgaven er altså:

$$min \qquad \sum_{j\in V}\sum_{i\in I_j}\sum_{k\in K_j}\sum_{u\in U} C^V x^k_{iju} + \sum_{j\in N}\sum_{i\in I_j}\sum_{k\in K_j}\sum_{u\in U} C^{TT} K_{MTT_{ij}} x^k_{iju} \tag{4.23}$$

$$\sum_{i\in I_j} x^k_{iju} - \sum_{i\in O_j} x^k_{jiu} = 0 \qquad \forall j \in N, k \in K_j, u \in U \tag{4.24}$$

$$\sum_{k\in K_j}\sum_{i\in I_j} x^k_{iju} = 1 \qquad \forall j \in D, u \in U \tag{4.25}$$

$$\sum_{k\in K_j}\sum_{i\in I_j} x^k_{iju} \leq 1 \qquad \forall j \in V, u \in U \tag{4.26}$$

[1] I vedlegg E er det presentert resultater for en modell løst med XpressMP der dette kravet er inkludert.

$$z^k_{iu} - z^k_{ju} + M1_{ij}x^k_{iju} \leq M1_{ij} - KMTT_{ij} - LENGDE_j \quad \forall j \in D, i \in D \cap I_j, k \in K_j, u \in U \tag{4.27}$$

$$z^k_{iu} + M2_{ij}x^k_{iju} \leq M2_{ij} - KMTT_{ij} + KMGRENSE^k \quad \forall j \in V, i \in I_j, k \in K_j, u \in U \tag{4.28}$$

$$M3_{ij}x^k_{iju} - z^k_{ju} \leq M3_{ij} - KMTT_{ij} - LENGDE_j \quad \forall j \in D, i \in V \cap I_j, k \in K_j, u \in U \tag{4.29}$$

$$\sum_{j \in N | G_j = 1} \sum_{i \in I_j | G_i = T_{MAX}} x^k_{iju} \leq MATKAP^k \qquad \forall k \in K, u \in U \tag{4.30}$$

$$\left(x^k_{iju} + \frac{1}{2} \sum_{l \in V \cap O_i \cap I_j} (x^k_{ilu} + x^k_{lju})\right) - \left(x^k_{ij(u-1)} + \frac{1}{2} \sum_{l \in V \cap O_i \cap I_j} (x^k_{il(u-1)} + x^k_{lj(u-1)})\right) = 0 \quad \forall j \in D, i \in D \cap I_j, k \in K_j, u \in (2, .., U_{MAX}) \tag{4.31}$$

$$0 \leq z^k_{ju} \leq KMGRENSE^k \qquad \forall j \in D, k \in K_j, u \in U \tag{4.32}$$

$$x^k_{iju} \in \{0, 1\} \qquad \forall j \in N, i \in I_j, k \in K_j, u \in U \tag{4.33}$$

4.3 Løsningsalgoritmer

Tre løsningsalgoritmer ble presentert i kapittel 3 og har blitt vurdert opp mot problemformuleringen til NSB. I dette delkapitlet presenteres den totale modellen (4.23) - (4.33) formulert med disse løsningsteknikkene.

For å forenkle problembeskrivelsene videre i dette kapitlet, har jeg erstattet målfunksjonen, (4.23) med følgende uttrykk:

$$f(x^k_{iju}) = \sum_{j \in V} \sum_{i \in I_j} \sum_{k \in K_j} \sum_{u \in U} C^V x^k_{iju} + \sum_{j \in N} \sum_{i \in I_j} \sum_{k \in K_j} \sum_{u \in U} C^{TT} KMTT_{ij} x^k_{iju} \tag{4.34}$$

4.3.1 Modell med Lagrangerelaksering

Hvis man studerer det totale problemet som ble formulert i forrige delkapittel, ser man at de fleste av restriksjonene er for hver uke og for hver materielltype. Man kan relaksere de restriksjonene som ikke er for hver uke og materielltype, slik at problemet kan løses per uke og per materielltype. Hvis man ser på problemstørrelsen, vil ikke dette problemet bli nevneverdig enklere å løse enn det opprinnelige problemet. Dette fordi det er få materielltyper i problemet, så for hver materielltype er det fremdeles et relativt stort problem å løse.

Derfor kan det være smart å relaksere noen av de andre restriksjonene i stedet. Ved å relaksere tellerrestriksjonene vil problemet ha denne strukturen:

$$
\begin{aligned}
min \quad & f(x^k_{iju}) \\
+ \quad & \sum_{j\in D}\sum_{i\in D\cap I_j}\sum_{k\in K_j}\sum_{u\in U}\alpha^k_{iju}\big(M1^k_{ij} - KMTT_{ij} - LENGDE_j - z^k_{iu} + z^k_{ju} - M1^k_{ij}x^k_{iju}\big) \\
+ \quad & \sum_{j\in V}\sum_{i\in I_j}\sum_{k\in K_j}\sum_{u\in U}\beta^k_{iju}\big(M2_{ij} - KMTT_{ij} + KMGRENSE^k - z^k_{iu} - M2_{ij}x^k_{iju}\big) \\
+ \quad & \sum_{j\in D}\sum_{i\in V\cap I_j}\sum_{k\in K_j}\sum_{u\in U}\gamma^k_{iju}\big(M3_{ij} - KMTT_{ij} - LENGDE_j - M3_{ij}x^k_{iju} + z^k_{ju}\big) \qquad (4.35)
\end{aligned}
$$

$$\sum_{i\in I_j} x^k_{iju} - \sum_{i\in O_j} x^k_{jiu} = 0 \qquad \forall j \in N, k \in K_j, u \in U \qquad (4.36)$$

$$\sum_{k\in K_j}\sum_{i\in I_j} x^k_{iju} = 1 \qquad \forall j \in D, u \in U \qquad (4.37)$$

$$\sum_{k\in K_j}\sum_{i\in I_j} x^k_{iju} \le 1 \qquad \forall j \in V, u \in U \qquad (4.38)$$

$$\sum_{i\in I_j|G_i=T_{MAX}}\ \sum_{j\in N|G_j=1} x^k_{iju} \le MATKAP^k \qquad \forall k \in K, u \in U \qquad (4.39)$$

$$\big(x^k_{iju} + \frac{1}{2}\sum_{l\in V\cap O_i\cap I_j}(x^k_{ilu} + x^k_{lju})\big) - \big(x^k_{ij(u-1)} + \frac{1}{2}\sum_{l\in V\cap O_i\cap I_j}(x^k_{il(u-1)} + x^k_{lj(u-1)})\big) = 0$$
$$\forall j \in D, i \in D\cap I_j, k \in K_j, u \in (2, ..., U_{MAX}) \qquad (4.40)$$

$$0 \le z^k_{ju} \le KMGRENSE^k \qquad \forall j \in D, k \in K_j, u \in U \qquad (4.41)$$

$$
\begin{aligned}
\alpha^k_{iju} &\ge 0 \qquad \forall j \in D, i \in D\cap I_j, k \in K_j, u \in U \qquad (4.42)\\
\beta^k_{iju} &\ge 0 \qquad \forall j \in V, i \in I_j, k \in K_j, u \in U \qquad (4.43)\\
\gamma^k_{iju} &\ge 0 \qquad \forall j \in D, i \in V\cap I_j, k \in K_j, u \in U \qquad (4.44)
\end{aligned}
$$

$$x^k_{iju} \in \{0,1\} \qquad \forall j \in N, i \in I_j, k \in K_j, u \in U \tag{4.45}$$

Hvis dette problemet (4.35)-(4.45) har heltallsegenskap, vil det ikke være noen god idé å relaksere disse restriksjonene. Da vil ikke den nedre grensen bli noe bedre enn om man lineærrelakserer hovedproblemet [Beasley, 1993]. Alle høyresidene til restriksjonene er heltall, men siden det er et flervareflytproblem, vet man ikke om akkurat dette problemet har heltallsegenskapen for x-variablene. Må se litt mer på problemet for å kunne se om denne relakseringen vil føre fram. I tillegg må man finne en måte å endre Lagrangemultiplikatorene på for hver iterasjon, kan for eksempel bruke subgradient optimization fra forrige kapittel. Et siste punkt er å se på hvordan løsningen ser ut når man er ferdig med Lagrangeiterasjonene. Problemet med å bruke Lagrangerelaksering er at når man finner en løsning på det Lagrangerelakserte problemet er det ikke sikkert at denne er den optimale løsningen for hele problemet. Dette kommer an på om restriksjonene som ble relaksert blir oppfylt. Hvis noen av disse ikke er oppfylt, må man transformere løsningen av Lagrangeproblemet slik at det blir en lovlig løsning til det opprinnelige problemet.

4.3.2 Modell med Dantzig-Wolfe dekomponering

Ved å legge restriksjonene som kobler materielltyper i masterproblemet vil hvert subproblem i Dantzig- Wolfe kunne løses for hver materielltype og for hver uke.

Restriksjonene til hvert subproblem gitt av k og u er da:

$$\sum_{i \in I_j} x^k_{iju} - \sum_{i \in O_j} x^k_{jiu} = 0 \qquad \forall j \in N \tag{4.46}$$

$$z^k_{iu} - z^k_{ju} + M1^k_{ij} x^k_{iju} \leq M1^k_{ij} - KMTT_{ij} - LENGDE_j \qquad \forall j \in D, i \in D \cap I_j \tag{4.47}$$

$$z^k_{iu} + M2_{ij} x^k_{iju} \leq M2_{ij} - KMTT_{ij} + KMGRENSE^k \qquad \forall j \in V, i \in I_j \tag{4.48}$$

$$M3_{ij} x^k_{iju} - z^k_{ju} \leq M3_{ij} - KMTT_{ij} - LENGDE_j \qquad \forall j \in D, i \in V \cap I_j \tag{4.49}$$

$$\sum_{i|G_i=T_{MAX}} \sum_{j|G_j=1} x^k_{iju} \leq MATKAP^k \tag{4.50}$$

$$\Big(x^k_{iju} + \frac{1}{2} \sum_{l \in V \cap O_i \cap I_j} (x^k_{ilu} + x^k_{lju})\Big) - \Big(x^k_{ij(u-1)} + \frac{1}{2} \sum_{l \in V \cap O_i \cap I_j} (x^k_{il(u-1)} + x^k_{lj(u-1)})\Big) = 0 \qquad \forall j \in D, i \in D \cap I_j \tag{4.51}$$

$$0 \leq z_{ju}^{k} \leq KMGRENSE^{k} \qquad \forall j \in D \tag{4.52}$$

$$x_{iju}^{k} \in \{0,1\} \qquad \forall j \in N, i \in I_j \tag{4.53}$$

Hvis man ser på subproblemene i Dantzig-Wolfe, så har ikke problemet blitt så mye enklere å løse. Siden det er et begrenset antall materielltyper, vil ikke subproblemene bli mye enklere å løse enn det totale problemet. I dette problemet har vi kun 3 materielltyper, og subproblemene vil fremdeles ha mange beslutningsvariable hvis denne løsningsstrategien brukes, det vil si at subproblemene ikke er vesentlig redusert i forhold til å løse problemet uten Dantzig-Wolfe dekomponering, samtidig som subproblemet blir et heltallsproblem.

4.3.3 Modell med Benders dekomponeringsalgoritme

Problemet er et blandet heltallsproblem. Ved å dekomponere problemet med Benders dekomponeringsteknikk, skiller man variablene slik at en variabel hører til hvert problem.

For en gitt verdi på x-variablene reduseres hovedproblemet fra kapittel 4.2 til et lineært subproblem:

$$min \qquad f(\bar{x}_{iju}^{k}) + \sum_{j \in D} \sum_{k \in K_j} \sum_{u \in U} z_{ju}^{k} \tag{4.54}$$

$$z_{iu}^{k} - z_{ju}^{k} \leq M1_{ij}^{k} - KMTT_{ij} - LENGDE_j - M1_{ij}^{k} \bar{x}_{iju}^{k} \qquad \forall j \in D, i \in D \cap I_j, k \in K_j, u \in U \tag{4.55}$$

$$z_{iu}^{k} \leq M2_{ij} - KMTT_{ij} + KMGRENSE^{k} - M2_{ij} \bar{x}_{iju}^{k} \qquad \forall j \in V, i \in I_j, k \in K_j, u \in U \tag{4.56}$$

$$-z_{ju}^{k} \leq M3_{ij} - KMTT_{ij} - LENGDE_j - M3_{ij} \bar{x}_{iju}^{k} \qquad \forall j \in D, i \in V \cap I_j, k \in K_j, u \in U \tag{4.57}$$

$$z_{ju}^{k} \leq KMGRENSE^{k} \qquad \forall j \in D, k \in K_j, u \in U \tag{4.58}$$

$$z_{ju}^{k} \geq 0 \qquad \forall j \in D, k \in K_j, u \in U \tag{4.59}$$

Hvis man ser på problemet (4.54) - (4.59) ser dette ganske enkelt ut å løse. Det er separabelt med hensyn på materielltype og uke. Restriksjonen i problemet setter grensene til tellevariablene. Differansen mellom to variable settes av restriksjonen (4.55). Problemet setter tellervariablene på den nederste grensen på grunn av målfunksjonen.

Her har jeg valgt å minimere tellerne siden koblingene mellom dagsløpene allerede er gitt.

For å forenkle modellene litt, vil jeg i resten av kapitlet bruke følgende uttrykk:

$$g1^k_{iju}(\bar{x}^k_{iju}) = M1^k_{ij} - KMTT_{ij} - LENGDE_j - M1^k_{ij}\bar{x}^k_{iju} \quad (4.60)$$

$$g2^k_{iju}(\bar{x}^k_{iju}) = M2_{ij} - KMTT_{ij} + KMGRENSE^k - M2_{ij}\bar{x}^k_{iju} \quad (4.61)$$

$$g3^k_{iju}(\bar{x}^k_{iju}) = M3_{ij} - KMTT_{ij} - LENGDE_j - M3_{ij}\bar{x}^k_{iju} \quad (4.62)$$

Dualen til (4.54)-(4.59) er også et lineært subproblem:

$$\begin{aligned} max \quad & f(\bar{x}^k_{iju}) \\ + \quad & \sum_{j\in D}\sum_{i\in D\cap I_j}\sum_{k\in K_j}\sum_{u\in U} g1^k_{iju}(\bar{x}^k_{iju})\alpha^k_{iju} \\ + \quad & \sum_{j\in V}\sum_{i\in I_j}\sum_{k\in K_j}\sum_{u\in U} g2^k_{iju}(\bar{x}^k_{iju})\beta^k_{iju} \\ + \quad & \sum_{j\in D}\sum_{i\in V\cap I_j}\sum_{k\in K_j}\sum_{u\in U} g3^k_{iju}(\bar{x}^k_{iju})\gamma^k_{iju} \\ + \quad & \sum_{j\in D}\sum_{k\in K_j}\sum_{u\in U} KMGRENSE^k\delta^k_{ju} \end{aligned} \quad (4.63)$$

$$\alpha^k_{iju} + \delta^k_{ju} \leq 1 \qquad \forall j \in D, i \in D \cap I_j, k \in K_j, u \in U \quad (4.64)$$

$$-\alpha^k_{iju} + \delta^k_{ju} \leq 1 \qquad \forall j \in D, i \in D \cap I_j, k \in K_j, u \in U \quad (4.65)$$

$$-\gamma^k_{iju} + \delta^k_{ju} \leq 1 \qquad \forall j \in D, i \in V \cap I_j, k \in K_j, u \in U \quad (4.66)$$

$$\beta^k_{iju} \leq 1 \qquad \forall j \in V, i \in I_j, k \in K_j, u \in U \quad (4.67)$$

$$\alpha^k_{iju} \leq 0 \qquad \forall j \in D, i \in D \cap I_j, k \in K_j, u \in U \quad (4.68)$$

$$\beta^k_{iju} \leq 0 \qquad \forall j \in V, i \in I_j, k \in K_j, u \in U \quad (4.69)$$

$$\gamma^k_{iju} \leq 0 \qquad \forall j \in D, i \in V \cap I_j, k \in K_j, u \in U \quad (4.70)$$

$$\delta^k_{ju} \leq 0 \qquad \forall j \in D, k \in K_j, u \in U \quad (4.71)$$

Benders masterproblem er (følger fremgangsmåten som er vist i kapittel 3.3):

$$
\begin{aligned}
min_{x^k_{iju}} \quad & y && (4.72)\\
y \quad & \geq f(x^k_{iju}) \\
+ \quad & \sum_{j\in D}\sum_{i\in D\cap I_j}\sum_{k\in K_j}\sum_{u\in U} g1^k_{iju}(x^k_{iju})\alpha^k_{iju} \\
+ \quad & \sum_{j\in V}\sum_{i\in I_j}\sum_{k\in K_j}\sum_{u\in U} g2^k_{iju}(x^k_{iju})\beta^k_{iju} \\
+ \quad & \sum_{j\in D}\sum_{i\in V\cap I_j}\sum_{k\in K_j}\sum_{u\in U} g3^k_{iju}(x^k_{iju})\gamma^k_{iju} \\
+ \quad & \sum_{j\in D}\sum_{k\in K_j}\sum_{u\in U} KMGRENSE^k\delta^k_{ju} \qquad \forall(\alpha,\beta,\gamma,\delta)\in P_{\mathbf{D}} && (4.73)
\end{aligned}
$$

$$\sum_{i\in I_j} x^k_{iju} - \sum_{i\in O_j} x^k_{jiu} = 0 \qquad \forall j\in N, k\in K_j, u\in U \tag{4.74}$$

$$\sum_{k\in K_j}\sum_{i\in I_j} x^k_{iju} = 1 \qquad \forall j\in D, u\in U \tag{4.75}$$

$$\sum_{k\in K_j}\sum_{i\in I_j} x^k_{iju} \leq 1 \qquad \forall j\in V, u\in U \tag{4.76}$$

$$\sum_{i\in I_j|G_i=T_{MAX}}\ \sum_{j\in N|G_j=1} x^k_{iju} \leq MATKAP^k \qquad \forall k\in K, u\in U \tag{4.77}$$

$$\Big(x^k_{iju} + \frac{1}{2}\sum_{l\in V\cap O_i\cap I_j}(x^k_{ilu}+x^k_{lju})\Big) - \Big(x^k_{ij(u-1)} + \frac{1}{2}\sum_{l\in V\cap O_i\cap I_j}(x^k_{il(u-1)}+x^k_{lj(u-1)})\Big) = 0$$
$$\forall j\in D, i\in D\cap I_j, k\in K_j, u\in(2,...,U_{MAX}) \tag{4.78}$$

$$
\begin{aligned}
& \sum_{j\in D}\sum_{i\in D\cap I_j}\sum_{k\in K_j}\sum_{u\in U} g1^k_{iju}(x^k_{iju})\alpha^k_{iju} \\
+ \ & \sum_{j\in V}\sum_{i\in I_j}\sum_{k\in K_j}\sum_{u\in U} g2^k_{iju}(x^k_{iju})\beta^k_{iju} \\
+ \ & \sum_{j\in D}\sum_{i\in V\cap I_j}\sum_{k\in K_j}\sum_{u\in U} g3^k_{iju}(x^k_{iju})\gamma^k_{iju} \\
+ \ & \sum_{j\in D}\sum_{k\in K_j}\sum_{u\in U} KMGRENSE^k\delta^k_{ju} \\
\leq \ & 0 \qquad \forall(\alpha,\beta,\gamma,\delta)\in Q_{\mathbf{D}} && (4.79)
\end{aligned}
$$

$$x^k_{iju}\in\{0,1\} \qquad \forall j\in N, i\in I_j, k\in K_j, u\in U \tag{4.80}$$

(4.73) er optimalitetskuttene og (4.79) er lovlighetskuttene.

Hvis man studerer masterproblemet og subproblemet til Benders algoritme, ser man at masterproblemet vil være et flervareflyt i nettverksproblem, der alle dagsløpsnodene skal dekkes. Subproblemet vil sjekke om sammenkoblingene som blir gjort av dagsløpene i masterproblemet er lovlige med hensyn på vedlikeholdsintervallene. Når man ser nærmere på subproblemet, er målfunksjonen å minimere telleren. Den ble kun lagt til problemet for å legge tellervariablene så lavt som mulig, og ikke på sin øvre grense. Derfor vil det, når det har blitt en lovlig løsning i subproblemet ikke være noe poeng å kjøre algoritmen mer, siden man da har en optimal kobling, og de koblingene er lovlige. Dette vil avvike litt fra Benders dekomponeringsalgoritme, siden den går ut på å løse masterproblemet og subproblemet etter hverandre helt til målfunksjonsverdiene er like. Den totale kjørelengden blir minimert siden tomtogene er med i målfunksjonen til masterproblemet.

Siden man kan fjerne optimalitetskuttene fra masterproblemet, vil Benders masterproblem se slik ut:

$$min_{x^k_{iju}} f(x^k_{iju}) \tag{4.81}$$

$$\sum_{i\in I_j} x^k_{iju} - \sum_{i\in O_j} x^k_{jiu} = 0 \qquad \forall j \in N, k \in K_j, u \in U \tag{4.82}$$

$$\sum_{k\in K_j}\sum_{i\in I_j} x^k_{iju} = 1 \qquad \forall j \in D, u \in U \tag{4.83}$$

$$\sum_{k\in K_j}\sum_{i\in I_j} x^k_{iju} \leq 1 \qquad \forall j \in V, u \in U \tag{4.84}$$

$$\sum_{i\in I_j|G_i=T_{MAX}}\ \sum_{j\in N|G_j=1} x^k_{iju} \leq MATKAP^k \qquad \forall k \in K, u \in U \tag{4.85}$$

$$\Big(x^k_{iju} + \frac{1}{2}\sum_{l\in V\cap O_i\cap I_j}(x^k_{ilu} + x^k_{lju})\Big) - \Big(x^k_{ij(u-1)} + \frac{1}{2}\sum_{l\in V\cap O_i\cap I_j}(x^k_{il(u-1)} + x^k_{lj(u-1)})\Big) = 0$$
$$\forall j \in D, i \in D\cap I_j, k \in K_j, u \in (2,...,U_{MAX}) \tag{4.86}$$

$$\begin{aligned}
&\sum_{j\in D}\sum_{i\in D\cap I_j}\sum_{k\in K_j}\sum_{u\in U} g1^k_{iju}(x^k_{iju})\alpha^k_{iju} \\
+\ &\sum_{j\in V}\sum_{i\in I_j}\sum_{k\in K_j}\sum_{u\in U} g2^k_{iju}(x^k_{iju})\beta^k_{iju} \\
+\ &\sum_{j\in D}\sum_{i\in V\cap I_j}\sum_{k\in K_j}\sum_{u\in U} g3^k_{iju}(x^k_{iju})\gamma^k_{iju} \\
+\ &\sum_{j\in D}\sum_{k\in K_j}\sum_{u\in U} KMGRENSE^k\delta^k_{ju} \\
\leq\ &0 \qquad \forall(\alpha,\beta,\gamma,\delta) \in Q_{\mathbf{D}}
\end{aligned} \tag{4.87}$$

$$x^k_{iju} \in \{0,1\} \qquad \forall j \in N, i \in I_j, k \in K_j, u \in U \tag{4.88}$$

Fase1-problem kan brukes for å generere lovlighetskutt, og algoritmen som kan implementeres for NSB sitt problem er derfor:

Algoritme 4.1 *Algoritme for å løse vedlikeholdsproblemet.*

1. *Sett $t = 1$ og mengden med ekstremalpunkt, $P^1_{\boldsymbol{D}}$ og mengden med stråler, $Q^1_{\boldsymbol{D}}$ tom.*
2. *Løs Benders masterproblem, problem (4.81)-(4.88).*
   (a) *Hvis masterproblemet er uløselig, vil det originale problemet også være uløselig. Stopp algoritmen.*
   (b) *Ellers, la $\bar{x}^t$ være en optimal løsning med målfunksjonsverdien LB^t (som vil være en nedre grense på det originale problemet's verdi)*
3. *Løs Fase1-problemet.*
   (a) *Hvis målfunksjonsverdien er større enn null, vil subproblemet være uløselig, lag et lovlighetskutt. La $(\alpha, \beta, \gamma, \delta)^t$ være en ekstrem stråle slik at*

$$\begin{aligned} & \sum_{j\in D}\sum_{i\in D\cap I_j}\sum_{k\in K_j}\sum_{u\in U} g1^k_{iju}(x^k_{iju})\alpha^k_{iju} \\ + & \sum_{j\in V}\sum_{i\in I_j}\sum_{k\in K_j}\sum_{u\in U} g2^k_{iju}(x^k_{iju})\beta^k_{iju} \\ + & \sum_{j\in D}\sum_{i\in V\cap I_j}\sum_{k\in K_j}\sum_{u\in U} g3^k_{iju}(x^k_{iju})\gamma^k_{iju} \\ + & \sum_{u\in U}\sum_{k\in K_j}\sum_{j\in D} \textsc{Kmgrense}^k \delta^k_{ju} \\ > & \ 0 \qquad \forall(\alpha,\beta,\gamma,\delta) \in Q_{\boldsymbol{D}} \end{aligned}$$

   Lag et lovlighetskutt ved å sette $Q^{t+1}_{\boldsymbol{D}} = Q^t_{\boldsymbol{D}} \cup \{(\alpha, \beta, \gamma, \delta)^t\}$. Sett $P^{t+1}_{\boldsymbol{D}} = P^t_{\boldsymbol{D}}$. Sett $t = t+1$ og gå til trinn 2.
   (b) *Hvis målfunksjonsverdien er null, vil subproblemet være lovlig. Gå til trinn 4 og løs subproblemet.*
4. *Løs det primale subproblemet. Hvis det duale subproblemet er begrenset/har endelig løsning, la z^t være en primal optimal løsning med målfunksjonsverdi UB^t. Stopp algoritmen siden man nå har et optimalt masterproblem og et lovlig subproblem.*

Fase1-problemet til subproblemet vil være:

$$min \sum_{u\in U}\sum_{k\in K}\left(\sum_{j\in D}\sum_{i\in I_j\cap D} v1_{iju} + \sum_{j\in V}\sum_{i\in I_j} v2^k_{iju} + \sum_{j\in D}\sum_{i\in I_j\cap V} v3^k_{iju} + \sum_{j\in D} v4^k_{ju}\right) \tag{4.89}$$

$$z^k_{iu} - z^k_{ju} - v1^k_{iju} \leq g1^k_{iju}(\bar{x}^k_{iju}) \qquad \forall j \in D, i \in D \cap I_j, k \in K_j, u \in U \tag{4.90}$$

$$z^k_{iu} - v2^k_{iju} \leq g2^k_{iju}(\bar{x}^k_{iju}) \qquad \forall j \in V, i \in I_j, k \in K_j, u \in U \tag{4.91}$$

$$-z^k_{ju} - v3^k_{iju} \leq g3^k_{iju}(\bar{x}^k_{iju}) \qquad \forall j \in D, i \in V \cap I_j, k \in K_j, u \in U \tag{4.92}$$

$$z^k_{ju} - v4^k_{ju} \leq KMGRENSE^k \qquad \forall j \in D, k \in K_j, u \in U \tag{4.93}$$

$$\begin{aligned}
z^k_{ju} &\geq 0 \qquad \forall j \in D, k \in K_j, u \in U && (4.94)\\
v1^k_{iju} &\geq 0 \qquad \forall j \in D, i \in I_j \cap D, k \in K_j, u \in U && (4.95)\\
v2^k_{iju} &\geq 0 \qquad \forall j \in V, i \in I_j, k \in K_j, u \in U && (4.96)\\
v3^k_{iju} &\geq 0 \qquad \forall j \in D, i \in I_j \cap V, k \in K_j, u \in U && (4.97)\\
v4^k_{ju} &\geq 0 \qquad \forall j \in D, k \in K_j, u \in U && (4.98)
\end{aligned}$$

Har lagt til kunstvariabler til hver restriksjon, og så lenge disse kunstvariablene har verdi, vil ikke det primale subproblemet ha lovlig løsning. Da må det genereres lovlighetskutt, og man må løse masterproblemet på nytt.

4.4 Valg av løsningsalgoritme

Bender og Dantzig-Wolfe er to løsningsalgoritmer som vil lede til optimalitet. Lagrangerelaksering kan føre til optimalitet, men dette er avhengig av hva som skjer med restriksjonene som er relaksert. Det kan være vanskelig å transformere en optimal Lagrangeløsning til en lovlig løsning for hele problemet. Samtidig må man lage en algoritme for hvordan Lagrangemultiplikatorene skal endre seg i hver iterasjon. En algoritme basert på Lagrangerelaksering har ingen stoppregel eller fører nødvendigvis ikke til forbedring i løsningen hver gang.

I utgangspunktet er masterproblemet i Bender nesten det samme som det Lagrangerelakserte problemet, (4.35)-(4.45). Hvis dette problemet har heltallsegenskap, vil det bli lett å løse med Benders formulering. Hvis problemet ikke har heltallsegenskapen kan det fremdeles løses med Bender, men det kan også løses med Lagrangerelaksering siden den nedre grensen da vil være bedre enn om man kun lineærrelakserer problemet.

Ved å studere litteraturen som er gjort på området er det lite litteratur skrevet på vedlikeholdsproblemet. Finner ingen modell som har det samme fokuset som modellen i dette arbeidet. Gjennom litteraturstudiet har jeg derfor ikke funnet ut at en løsningsalgoritme vil passe bedre enn en annen på dette problemet, og jeg kan derfor ikke si at den ene løsningsalgoritmen vil være helt sikkert bedre enn en annen for dette problemet. Men det ser ut som om Dantzig-Wolfe ikke vil føre fram, siden det blir store subproblemer med binærkrav. Jeg kan ikke trekke en konklusjon på hva som er den

beste algoritmen av Lagrangerelaksering og Benders dekomponering uten å implementere og teste med begge for dette problemet. På grunn av begrenset med tid, velger jeg kun å implementere en algoritme. Masterproblemet til Bender og det Lagrangerelakserte problemet ser ganske like ut, og er sannsynligvis omtrent like lette å løse. Siden subproblemet til Bender er enkelt å løse, og i tillegg Benders algoritme garanterer optimalitet, har jeg valgt å implementere en variant av Benders algoritme.

I følge Magnanti and Wong [1981] kan en implementasjon av Benders algoritme rett ut føre til sakte konvergens for nettverksproblemer, og kreve løsning av et stort antall heltallsproblemer. I litteraturstudiet har jeg sett på noen planleggingsproblemer som bruker Benders dekomponering for å løse nettverksproblemer, og jeg har funnet tips til hvordan algoritmen kan konvergere raskere fra Cordeau et al. [2000, 2001b], McDaniel and Devine [1977] og Magnanti and Wong [1981]. Dette vil jeg komme tilbake til i neste kapittel.

Kapittel 5

Implementering og testing

Problemet har blitt implementert i en kommersiell programvare, XpressMP. Jeg har valgt å bruke XpressMP siden det er den programvaren jeg har brukt i tidligere fag på NTNU. XpressMP ble også brukt som optimeringsverktøy i Bjorvand and Risberg [2004]. XpressMP benytter seg blant annet av løsningsalgoritmene simplex og Branch & Bound for å løse problemene. Versjonene som har blitt brukt er Xpress-IVE versjon 1.15.04, Xpress Mosel Versjon 1.4.1 og Xpress Optimizer Versjon 15.25.03. Datamaskinen som modellene har kjørt på er en Pentium 4 1,5 GHz prosessor med 768 MB minne. Jeg kan ikke garantere at det ikke har blitt kjørt andre prosessorkrevende jobber på maskinen under kjøring og testing. I dette kapitlet vil jeg først kommentere implementasjonen av modellen før jeg vil kommentere resultatene etter å ha kjørt og testet modellen.

5.1 Implementering

Jeg har implementert problemet i XpressMP på to måter. En ved å formulere hele problemet fra kapittel 4.2 i en Mosel-fil i XpressMP. Denne implementeringen har jeg heretter kalt *LØS1*, og denne formuleringen bygger litt videre på det som ble gjort i Bjorvand and Risberg [2004]. For kommentarer til implementeringen og løsning av denne modellen henviser jeg til Bjorvand and Risberg [2004].

I denne oppgaven har jeg ikke prioritert å se noe nærmere på hvordan man kan få kortere løsningstid med denne formuleringen, siden det i Bjorvand and Risberg [2004] ble konkludert med at man måtte løse problemet med en mer effektiv løsningsalgoritme enn kun å formulere problemet i XpressMP. Formuleringen i XpressMP er derfor tatt med i dette arbeidet for å ha noe å sammenlikne den implementerte løsningsalgoritmen med.

Den andre formuleringen har benyttet Benders formulering sammen med XpressMP, heretter kalt *Benders algoritme*. Benders masterproblem, fase 1 problem og subproblem har blitt modellert i hver sin Mosel fil og det er laget en batch fil som kjører disse modellene etter algoritme 4.1 i kapittel 4.3. Modellene som blir kjørt i XpressMP returnerer en kode når de har kjørt ferdig (exitkode), og batch scriptet er laget slik at det bestemmer hvilken modell som skal kjøres avhengig av hvilken exitkode forrige modell returnerte. .dat-filer benyttes til å lagre og hente data mellom modellene. Jeg valgte å implementere algoritmen i batch-kode, framfor i et eget programmeringsspråk etter

samtaler med veilederne mine ved NTNU og SINTEF. Ved å implementere algoritmen på denne måten kan mye av kildekoden i Mosel gjenbrukes, og det vil derfor være tidsbesparende å implementere algoritmen på denne måten. En av de største ulempene med å implementere det på denne måten er at for hver iterasjon må problemene "tegnes" på nytt. Algoritmen husker ikke restriksjoner fra iterasjon til iterasjon, og hele problemet må lastes inn på nytt i hver iterasjon. Fokuset i dette arbeidet har vært å få en løsning av problemet med algoritmen framfor den mest effektive programmeringsmetoden.

5.1.1 Benders algoritme

Etter å ha implementert Benders algoritme rett ut og sammenliknet løsningstidene med løsningstider for LØS1, var det et par trender som viste seg. Algoritmen bruker mye tid på å oppdatere datafiler mellom iterasjonene. Omtrent 30% av tida brukes til å oppdatere datafilene med lovlighetskutt[1] for de datasettene som algoritmen må kjøre flere iterasjoner for å løse. For de større datasettene, for eksempel typer8, der det må genereres mange lovlighetskutt er Xpress raskere til å løse problemet. Når antall lovlighetskutt øker, øker kjøretiden per masterproblem.

Mange iterasjoner kan tyde på at det er sakte konvergens med Benders algoritme som fører til at algoritmen må kjøre mange iterasjoner for å finne en optimal løsning for de datasettene som er litt verre å løse.

For å få algoritmen til å gå raskere er det dermed minst tre måter å forbedre løsningstiden på:

- Løse masterproblemet mer effektivt for å få raskere løsningstid per masterproblem. Siden masterproblemet er et heltalls flervareflyt i nettverksproblem kan man se på ulike nettverksalgoritmer som kan effektivisere løsningen av problemet. Samtidig kan man se på muligheter for å redusere mengden med lovlightskutt slik at antallet restriksjoner i masterproblemet blir mindre, og det blir mindre problem å laste.

- Raskere konvergens for Benders algoritme, slik at antall iterasjoner kan reduseres.

- Forbedre oppdatering i datafilene.

Videre i dette delkapitlet presenteres ulike metoder for å løse Benders masterproblem mer effektiv, og for å få Benders algoritme til å konvergere raskere. Jeg har valgt å ikke fokusere på å få oppdateringen av lovlighetskuttene i fil mer effektiv fordi dette i hovedsak har med valg av programmeringsmetode å gjøre, og jeg har tatt et valg på implementasjonsmetode.

Det er flere muligheter for forbedring for konvergensen til Benders algoritme i følge Magnanti and Wong [1981]:

- Lage et godt utvalg med initielle kutt, for eksempel sette gode verdier på dualvariablene fra subproblemet når man skal løse masterproblemet. Magnanti and Wong [1981] refererer til et studie der det ble funnet ut at startutvalget med initielle kutt kunne ha en bra effekt på ytelsen til Benders algoritme.

[1] Testingen har foregått på en datamaskin med gammel harddisk, så dette kan være noe av årsaken til at å skrive lovlighetskuttene til fil har gått litt sakte.

- Formulere problemet ordentlig og hensiktsmessig.

- Hvis det er valg, velge gode kutt å legge til masterproblemet i hvert trinn.

Flere kutt per masterproblem

I følge Cordeau et al. [2000] kan det være smart å forbedre worst-case tilfellet til Benders algoritme. I verste fall må man kjøre like mange iterasjoner som det er ekstreme stråler til subproblemet. Algoritmen kan forbedres ved å dele opp subproblemet slik at hvert subproblem løses for hver uke og for hver materielltype. Da vil flere kutt kunne bli lagt til masterproblemet per iterasjon. I vedlikeholdsproblemet har man begrenset med materielltyper, så dette vil ikke redusere antall iterasjoner altfor mye. Fase-1 problemet er enkelt å løse, så det er implementert slik at det løser problemet med alle materielltypene og ukene i en modell. Fra fase1-problemet genereres det lovlighetskutt for hver uke og for hver materielltype til masterproblemet.

Initielle kutt

Det er også påpekt av Cordeau et al. [2000] at det kan være smart å legge til en del initielle kutt i forhold til løsningstiden. Man ønsker å tilføre noen kutt som beskjærer løsningsrommet dramatisk. Dette kan gjøres både ved initielle kutt og ved gode kuttvalg i hver iterasjon.

Cordeau et al. [2000] valgte å legge inn initielle optimalitetskutt, men å la mengden med ekstreme stråler være tom da de begynte å kjøre algoritmen. I dette vedlikeholdsproblemet er det viktigere å sette lovlighetskutt, siden algoritmen terminerer så lenge subproblemet er løselig. Jeg er kun interessert i at tellerne legger seg på sin nedre grense når det er en optimal løsning og det genereres derfor ingen optimalitetskutt.

Spørsmålet er hva slags lovlighetskutt som kan legges til i løsningen. De løsningene som ikke er lovlige er de syklene som ikke inneholder vedlikehold. Dette sjekker restriksjonene i subproblemet, og man kan vanskelig legge til restriksjoner til masterproblemet som sjekker dette uten å bruke variablene fra subproblemet.

Et lovlighetskutt man kan legge til er at antall vedlikehold må være større enn summen av alle dagsløpslengdene delt på kilometergrensen for problemet, samtidig som antall vedlikehold må være et heltall.

$$Vedlikehold \geq \sum_{k \in K} \lceil U * \sum_{j \in D} \frac{LENGDE_j}{KMGRENSE^k} NODETYPE_j^k \rceil \tag{5.1}$$

Når man ser på dette kuttet[2], har jeg sagt at antall vedlikehold må være et heltall. Dette minker løsningstiden. I tillegg må man gjøre det til et heltall per togtype da dette vil redusere løsningstiden for flertyperproblemer. Siden en vedlikeholdsnode ikke kan være innode eller utnode til en annen vedlikeholdsnode, vil vedlikeholdene automatisk spres litt i turneringsplanen som blir generert, og dette gjør løsningsrommet noe mindre.

Når man studerer dette resultatet virker det som om jo mer man kan si om løsningen før man begynner, desto kortere løsningstid blir det. Hvis total kjørelengde er 31000km,

[2]Merk at dette kuttet bare virker så lenge kun en materielltype kan kjøre hvert dagsløp. Hvis det er valg mellom flere materielltyper per dagsløp, må dette kuttet omformuleres litt.

er det behov for fire vedlikehold hvis kilometergrensen er 10000km. Det betyr at uansett hvordan man kobler, så må man bruke minst fire vedlikehold.

Hvis total kjørelengde i stedet for er 38000km, betyr det at antall vedlikehold er det samme, men det er færre lovlige muligheter til å koble dagsløpene sammen, siden det vil være mindre "slakk" i modellen. De fleste intervallene mellom to vedlikehold må fylles nesten opp for å kunne lage en plan med fire vedlikehold i dette tilfellet, og det vil som oftest ta lengre tid å finne en lovlig løsning.

I noen tilfeller kan den totale kjørelengden av dagsløp være veldig nært full utnyttelse av vedlikeholdsintervallene. Tomtogskjøringen kan være med på å øke antall nødvendige vedlikehold i dette tilfellet siden kjørelengden til kjøring av dagsløp og tomtog overstiger kjørelengden som var grunnlag for antallet vedlikehold som ble funnet i kutt (5.1) og gjør det nødvendig med et vedlikehold til.

Derfor er det viktig å finne en enda bedre nedre grense på problemet. En måte å gjøre dette på er at man finner en løsning av det lineærrelakserte totalproblemet. Fra dette problemet henter man tomtogskjøringen og får da en nedre grense til problemet som består av både den totale dagsløpslengden og tomtogskjøring. Dette vil gi en bedre nedre grense enn kun dagsløpslengden. Men denne nedre grenseverdien vil kun bli bedre for de problemene der tomtogskjøringen er med på å øke antall vedlikehold. For de andre datasettene har tomtogskjøringen ingenting å si, siden den nedre grensen vil være den samme.

Kuttformulering

I kapittel 3 ble prosedyren for Chvátal-Gomory (C-G) kutt presentert, og jeg har implementert lovlighetskuttene som blir generert fra fase1-problemet som C-G ulikheter. Disse kuttene vil være lovlige for alle heltallsproblemer, og de er nærmere heltallsløsning enn de genererte kuttene.

For å oversette C-G ulikhetene fra kapittel 3 til mitt problem har jeg negative dualverdier og kuttene er $\pi x \geq \pi_0$ i stedet for $\pi x \leq \pi_0$, så jeg kan dermed formulere de genererte lovlighetskuttene mine som C-G kutt, for å komme nærmere konvekse hull.

Relaksere masterproblemet

I følge McDaniel and Devine [1977] er en metode å legge til kutt til et heltallsproblem å starte algoritmen med å relaksere masterproblemet, for eksempel ved en LP-relaksering. En optimal løsning til heltallsproblemet vil også være en løsning til det relakserte problemet og ha samme målfunksjonsverdi. Lovlighetskutt vil ikke kappe bort løsninger som er lovlige i relakseringen, og dermed vil disse kuttene også være lovlige i masterproblemet.

I følge Cordeau et al. [2000] og McDaniel and Devine [1977] kan tilnærmingen implementeres på denne måten:

1. Løs LP-relakseringen av problemet ved å bruke algoritme 4.1.
2. Legg til heltallsrestriksjoner til Benders masterproblem
3. Restart Benders algoritme for å løse heltallsproblemet til optimalitet.

Hvis man kjører algoritmen på denne måten, vil det være en del kutt som blir lagt til når masterproblemet er relaksert og derfor enklere å løse. Kuttene som blir generert når problemet er lineærrelaksert vil sannsynligvis ikke være så stramme som om de hadde blitt generert av et heltallsproblem. Da går det an å se på metoder for å få disse ulikhetene nærmere konvekse hull. Både Schrijver [1993] og Nemhauser and Wolsey [1988] ser på hvordan ulikheter kan formuleres nærmere konvekse hull. For eksempel Schrijver [1993] strammer restriksjonene for hvert tog i planleggingsmodellen slik at restriksjonene tvinger problemområdet nærmere konvekse hull. I Nemhauser and Wolsey [1988] nevnes Gomorykutt som en måte å få kuttene nærmere konvekse hull.

Jeg har implementert to algoritmer, en som relakserer først og deretter løser heltallsproblemet og en algoritme som kun løser heltallsproblemet. Kuttene i det relakserte masterproblemet er også formulert som C-G ulikheter.

Fjerne kutt fra masterproblemet

I Cordeau et al. [2001b] har de implementert en tilnærming til pareto optimale kutt for de optimale kuttene i Benders algoritme. Man har to ekstremalpunkter u^1 og u^2 fra de tidligere eksemplet. De sier at et kutt generert fra ekstremalpunkt u^1 dominerer over et kutt fra ekstremalpunkt u^2 hvis $(b - g(y))^T u^1 \leq (b - g(y))^T u^2$, der $(b - g(y))^T u$ er målfunksjonen til subproblemet.

For algoritmen i denne oppgaven er det sett på muligheten for å finne dominerende kutt for lovlighetskuttene, siden modellen ikke har optimalitetskutt, og det er mange kutt som kutter av løsninger til masterproblemet.

En mulighet er å lage to lister, en med aktive og en med passive lovlighetskutt. Hvis et kutt ikke har vært aktivt i masterproblemet for et bestemt antall iterasjoner legges det i en passivliste og er ikke med i restriksjonssettet når man skal løse masterproblemet. Det må inkluderes en sjekk i ettertid som sjekker om løsningen vil være lovlig for de passive kuttene etter at man har løst problemet. Hvis kuttene ikke overholdes, må kuttene som gjør at løsingen er ulovlig legges til den aktive listen igjen. Denne strategien kunne blitt implementert om man også søkte en optimal løsning i subproblemet. Når man kun søker et lovlig subproblem, vil man ikke ha optimalitetskutt til å tvinge løsningen bort fra de første løsningene som blir funnet i masterproblemet. Når det fjernes lovlighetskutt som ikke har vært aktive i et visst antall iterasjoner, er det ingen garanti for at løsningen ikke kommer tilbake til en tidligere løsning. Og det kan bli mye bytting mellom den aktive og passive kuttlisten. Derfor har jeg valgt å ikke implementere en slik kuttreduserende algoritme.

Valget falt på å identifisere kutt som man vet er dominert av andre kutt, og fjerne disse kuttene fra restriksjonssettet. Da er det viktig å se på måter man kan holde rede på hvilke kutt som blir dominert av andre kutt, og fjerne kuttene fra den aktive listen med kutt. Man må altså sørge for at det er nok aktive kutt som sørger for at man ikke kommer tilbake til samme løsning i masterproblemet. I tillegg må man finne det minste antall kutt som beskriver restriksjonssettet med lovlighetskutt. Jeg har valgt å implementere en egen optimeringsmodell som vil fjerne kutt som blir dominert av andre kutt.

Siden dette er kutt der x skal være større enn noe (motsatt fra kapittel 3), vil et kutt, $\pi x \geq \pi_0$, dominere over et annet kutt, $\rho x \geq \rho_0$ hvis det eksisterer en $\mu > 0$ slik at:

$$\rho \leq \mu\pi \tag{5.2}$$
$$\rho_0 \geq \mu\pi_0 \tag{5.3}$$

Et lovlighetskutt ser slik ut:

$$\begin{aligned}
&\sum_{j\in D}\sum_{i\in D\cap I_j}\sum_{k\in K_j}\sum_{u\in U} g1^k_{iju}(x^k_{iju})\alpha^k_{iju} \\
+ &\sum_{j\in V}\sum_{i\in I_j}\sum_{k\in K_j}\sum_{u\in U} g2^k_{iju}(x^k_{iju})\beta^k_{iju} \\
+ &\sum_{j\in D}\sum_{i\in V\cap I_j}\sum_{k\in K_j}\sum_{u\in U} g3^k_{iju}(x^k_{iju})\gamma^k_{iju} \\
+ &\sum_{j\in D}\sum_{k\in K_j}\sum_{u\in U} K_{MGRENSE}{}^k\delta^k_{ju} \\
\leq\ & 0 \qquad \forall(\alpha,\beta,\gamma,\delta)\in Q_{\mathbf{D}}
\end{aligned} \tag{5.4}$$

Fra kapittel 4 har vi at:

$$g1^k_{iju}(x^k_{iju}) = M1^k_{ij} - K_{MTT_{ij}} - L_{ENGDE_j} - M1^k_{ij}x^k_{iju} \tag{5.5}$$
$$g2^k_{iju}(x^k_{iju}) = M2_{ij} - K_{MTT_{ij}} + K_{MGRENSE}{}^k - M2_{ij}x^k_{iju} \tag{5.6}$$
$$g3^k_{iju}(x^k_{iju}) = M3_{ij} - K_{MTT_{ij}} - L_{ENGDE_j} - M3_{ij}x^k_{iju} \tag{5.7}$$

Omformulerer kuttet (5.4) til:

$$\begin{aligned}
&\sum_{j\in D}\sum_{i\in D\cap I_j}\sum_{k\in K_j}\sum_{u\in U}(M1^k_{ij} - K_{MTT_{ij}} - L_{ENGDE_j})\alpha^k_{iju} \\
+ &\sum_{j\in V}\sum_{i\in I_j}\sum_{k\in K_j}\sum_{u\in U}(M2_{ij} - K_{MTT_{ij}} + K_{MGRENSE}{}^k)\beta^k_{iju} \\
+ &\sum_{j\in D}\sum_{i\in V\cap I_j}\sum_{k\in K_j}\sum_{u\in U}(M3_{ij} - K_{MTT_{ij}} - L_{ENGDE_j})\gamma^k_{iju} \\
+ &\sum_{j\in D}\sum_{k\in K_j}\sum_{u\in U} K_{MGRENSE}{}^k\delta^k_{ju} \\
\leq\ &\sum_{j\in D}\sum_{i\in D\cap I_j}\sum_{k\in K_j}\sum_{u\in U} M1^k_{ij}x^k_{iju}\alpha^k_{iju} \\
+ &\sum_{j\in V}\sum_{i\in I_j}\sum_{k\in K_j}\sum_{u\in U} M2_{ij}x^k_{iju}\beta^k_{iju} \\
+ &\sum_{j\in D}\sum_{i\in V\cap I_j}\sum_{k\in K_j}\sum_{u\in U} M3_{ij}x^k_{iju}\gamma^k_{iju} \\
&\forall(\alpha,\beta,\gamma,\delta)\in Q_{\mathbf{D}}
\end{aligned} \tag{5.8}$$

Hvis $l \in L$ er iterasjoner, så vil hver dualvariabel ha en indeks l som viser hvilken iterasjon kuttet ble generert. Variabelen $a_{kul_1l_2}$ er en binærvariabel og

$$a_{kul_1l_2} = \begin{cases} 1 & \text{hvis kuttet } l_2 \text{ er med i mengden som dominerer over kutt } l_1 \\ 0 & \text{ellers} \end{cases}$$

Problemet med å finne dominerende kutt vil være slik:

$$min \quad -\mu_{kul_1} + \sum_{l_2 \in L | l_1 \neq l_2} a_{kul_1l_2} \qquad \forall k \in K, u \in U, l_1 \in L \tag{5.9}$$

$$M1^k_{ij}\alpha^k_{ijul_1}\mu_{kul_1} \geq \sum_{l_2 \in L | l_1 \neq l_2} M1^k_{ij}\alpha^k_{ijul_2}a_{kul_1l_2} \qquad \forall j \in D, i \in D \cap I_j \tag{5.10}$$

$$M2_{ij}\beta^k_{ijul_1}\mu_{kul_1} \geq \sum_{l_2 \in L | l_1 \neq l_2} M2_{ij}\beta^k_{ijul_2}a_{kul_1l_2} \qquad \forall j \in V, i \in I_j \tag{5.11}$$

$$M3_{ij}\gamma^k_{ijul_1}\mu_{kul_1} \geq \sum_{l_2 \in L | l_1 \neq l_2} M3_{ij}\gamma^k_{ijul_2}a_{kul_1l_2} \qquad \forall j \in D, i \in V \cap I_j \tag{5.12}$$

$$\begin{aligned}
&\Bigg[\sum_{j \in D} K_{MGRENSE^k}\delta^k_{jul_1} \\
&+ \sum_{j \in D}\sum_{i \in I_j \cap D} \alpha^k_{ijul_2}(M1^k_{ij} - K_{MTT_{ij}} - L_{ENGDE_j}) \\
&+ \sum_{j \in V}\sum_{i \in I_j} \beta^k_{ijul_1}(M2_{ij} - K_{MTT_{ij}} + K_{MGRENSE^k}) \\
&+ \sum_{j \in D}\sum_{i \in V \cap I_j} \gamma^k_{ijul_1}(M3_{ij} - K_{MTT_{ij}} - L_{ENGDE_j})\Bigg]\mu_{kul_1} \\
&\leq \sum_{l_2 \in L | l_1 \neq l_2} \Bigg[\sum_{j \in D} K_{MGRENSE^k}\delta^k_{jul_2} \\
&+ \sum_{j \in D}\sum_{i \in I_j \cap D} \alpha^k_{ijul_2}(M1^k_{ij} - K_{MTT_{ij}} - L_{ENGDE_j}) \\
&+ \sum_{j \in V}\sum_{i \in I_j} \beta^k_{ijul_2}(M2_{ij} - K_{MTT_{ij}} + K_{MGRENSE^k}) \\
&+ \sum_{j \in D}\sum_{i \in V \cap I_j} \gamma^k_{ijul_2}(M3_{ij} - K_{MTT_{ij}} - L_{ENGDE_j})\Bigg]a_{kul_1l_2}
\end{aligned} \tag{5.13}$$

$$a_{kul_1l_2} \in \{0, 1\} \tag{5.14}$$

$$\mu_{kul_1} \geq 0 \tag{5.15}$$

Hvis problemet (5.9)-(5.15) er løselig og μ_{kul_1} har verdi større enn null, vil kuttet være dominert av en mengde med kutt der mengden kutt består av de som har variabelen $a_{kul_1l_2}$ lik en. Ut i fra denne modellen kan man altså fjerne kutt som blir dominert av andre kutt.

Jeg har ikke laget en sjekk som spesifikt sjekker om det er ekvivalente stråler. Det gjør det sannsynligvis ikke, siden jeg ikke har en initiell løsning med kutt, og siden

fase1-problemet ikke vil gi samme løsning. Hvis det finnes ekvivalente stråler vil disse bli fjernet fra løsningen i samme modell, siden ekvivalente stråler også vil bli identifisert som dominerende stråler.

Modellen som implementert tar lang tid å løse, og var derfor tenkt å kun kjøre for hver 30. eller hver 50. iterasjon. Kuttmodellen har en høy løsningstid fordi man for hvert kutt må sjekke om det finnes en mengde med kutt som dominerer dette kuttet. Ved å kjøre denne på datasett med 200 genererte lovlighetskutt, var det veldig få kutt som kunne fjernes fra løsningen. Siden så få kutt ble dominert, ble kravet til at et kutt er dominert gjort slakkere ved å kun teste en av likningene (5.2) og (5.3) om gangen. Dette førte heller ikke til at flere kutt ble dominert og kunne fjernes fra løsningen.

Siden så få kutt kunne bli fjernet fra løsningen er det ikke prioritert å finne den mest effektive metoden å fjerne kutt som blir dominert av andre kutt på. Hvis mange kutt kunne blitt fjernet, hadde jeg satt fokus på å finne en raskere og mer effektiv modell. For eksempel i Wallace and Wets [1993] identifiseres redundante Gale-Hoffman ulikheter[3] ved et enkelt kriterie.

Oppsummering - implementerte algoritmer

For å summere opp så er det implementert 3 versjoner av Benders algoritme, og disse er presentert i vedlegg B. De tre algoritmene har de samme initielle kuttene, lovlighetskuttene formulert som CG-kutt og sender flere kutt til masterproblemet per subproblem (per uke og type). Den første algoritmen tar kun hensyn til de elementene som er nevnt. Neste algoritme har i tillegg inkludert en sjekk for dominerende kutt, mens den siste har implementert en lineærrelaksering av masterproblemet først. I vedlegg C ligger en versjon av masterproblemet, fase1-problemet og subproblemet. For de ulike algoritmene vil exitkodene som modellene returnerer være forskjellig, siden batchscriptene er noe forskjellig.

5.2 Testing og resultater

Siden jeg stopper algoritmen når den første lovlige løsningen til subproblemet er funnet, vil det ikke være noen mulighet for å stoppe algoritmen tidligere for å få en lovlig løsning. Den optimale løsningen er den første lovlige løsningen som blir funnet. Ved å kjøre LØS1 kan man stoppe underveis når man har funnet en lovlig løsning siden XpressMP tar vare på siste beste løsning funnet. I dette kapitlet vil jeg presentere løsninger som har fremkommet ved LØS1 og ved Benders algoritme. I tabell 5.1 har jeg vist de ulike datasettene jeg har brukt som inndatasett til modellene.

Datasettene er laget ved å ta sykler fra materiellturneringsplanen til NSB i dag. Jeg har valgt ferdige sykler siden jeg da vet at det finnes løsninger. Man vil aldri rekke å teste alle mulige aspekter ved en modell i et slikt prosjekt, og jeg har derfor prioriert å lage datasett med varierende antall dagsløpsnoder og mulige vedlikehold. Alle de små datasettene har blitt generert ut i fra materiellturneringen til type 72. Datasett typer8

[3]Gale-Hoffman lemma karakteriserer ekstern flyt (supply, demand) i et rettet nettverk uttrykt ved et antall lineære ulikheter [Wallace and Wets, 1993].

er alle dagsløpene av togtype 72 på en av pendlene. De mindre datasettene som består av togtype 72 er også dagsløp hentet fra denne pendelen.

Navnene på datasettene er tilfeldig valgt, bortsett fra at typer72 er hele lokaltogsproblemet for materielltype 72, typer692 er hele problemet for materielltype 69-2, typer693 er hele problemet til materielltype 69-3 og typerALLE er hele problemet for lokaltog på Østlandet. Størrelsen på disse datasettene er også presentert i tabell 5.1.

I testingen har jeg brukt en modell som har lik kostnad for alle vedlikeholdene. For de små datasettene er det få vedlikeholdsnoder og dagsløpsnoder, og jeg har valgt vedlikehold på natten for å være sikker på at jeg får lovlige løsninger. For å få sammenliknbare resultater har jeg derfor satt lik kostnad på alle vedlikeholdene i alle modellene. Hvis man ønsker ulik kostnad på vedlikeholdsmulighetene kan dette enkelt endres i datasettene.

Tabell 5.1: Egenskaper for inndatasettene

Navn[c]	**DL node**	**VLH node**	**Egenskap**	**Togtype**	**VLH plan**[a]	**KM DL**[b]
typer1	14	14	Uten TT	Type72	1	8547.6
typer2	28	14	Uten TT	Type72	3	20609.4
typer3	28	14	Med TT	Type72	3	24004.5
typer4	28	14	Uten TT	Type72	3	23498.6
typer5	42	42	Med TT	Type69,72	5	31249.1
typer6	28	36	Uten TT	Type69,72	3	15575.7
typer7	42	14	Uten TT	Type72	4	32046.2
typer8	56	14	Uten TT	Type72	6	44108.0
typer72	105	42	Uten TT	Hele type72	10	73521.0
typer692	95	64	Uten TT	Hele type69-2	2	9895.1
typer693	326	64	Uten TT	Hele type69-3	16	115173.5
typerALLE	526	64	Uten TT	Alle	28	198590.1

a Dette er antall vedlikehold som ligger inne i planen til NSB.

b Lengden av dagsløpene i datasettene.

c Navnet på datasettene er tilfeldig valgt.

5.2.1 Flerukerproblemet

Hvis man studerer settene som blir løst for flere uker i tabell 5.2, er det et par resultater som må kommenteres for flerukerproblemet. Det er kortere løsningstid for det problemet som blir løst for ulike uker enn for like uker for inndatasett typer 2. Med like uker øker antall restriksjoner i problemet. Da vil problemet bli større å løse, siden en kobling er avhengig av en kobling i den påfølgende uken. Dette viser at denne restriksjonen gjør problemet vanskeligere å løse for flerukerproblemet. Videre i analysen har jeg valgt å se bort i fra flerukerproblemet, siden dette kun lar seg løse for de minste datasettene, og man får mindre problem når de er for en uke i stedet for flere uker. Samme resultat gjelder for flerukerproblemet med Benders algoritme.

Tabell 5.2: Resultater med LØS1 (formulering i Xpress).

Inndata	Uke[b]	Variabler[a]	Restr.[a]	Opt.	Løsetid	TT-km	VLH	LP[d]
typer1	1	120/95	157/126	ja	1.8s	325.1	1	1
typer1	2L	237/201	338/296	ja	1.5s	650.2	2	2
typer1	2U	237/209	310/279	ja	1.5s	650.2	2	2
typer2	1	276/271	327/324	ja	2.8s	768.4	3	3
typer2	2L	549/541	762/759	nei	3810.3s**	1536.8	6	5
typer2	2U	549/541	650/647	ja	83.3s	1534.8	5	5
typer3[c]	1	219/216	274/271	ja	1.8s	6.0	3	3
typer4	1	270/267	325/322	ja	1.9s	478.4	3	3
typer5	1	759/737	947/902	ja	15.1s	245.6	4	4
typer6	1	336/325	487/427	ja	2.4s	625.1	2	2
typer7	1	499/493	562/560	ja	10.1s	710.5	4	4
typer8	1	769/763	846/844	ja	714.2s	1109.4	5	5
typer72	1	3131/3081	3262/3262	nei	57693.7s**	2163.7	9	8
typer692	1	4038/3922	4129/4037	nei	62485.6s**	1022.1	2	1
typer692[e]	1	4038/3922	4129/4037	ja	91.0s	1022.1	2	2
typer693	1	23107/22811	23219/23037	nei	67608.0s**	5700.0	17	12
typerALLE	1	30273/29813	31574/30228	nei	72291.2**	12437.9	34	21

** Modellen ble stoppet før optimal løsning ble funnet.

a formulering/presolved. I Xpress er det innebygget presolvefunksjon.

b L = Like uker, U = Ulike uker

c TT = 0 (tomtog mellom noder ikke tillatt)

d Nedre grense for antall vedlikehold

e Satte den nedre grensen for problemet til 2 vedlikehold.

5.2.2 LØS1 - Formulering i Xpress

Resultatene fra LØS1 er vist i tabell 5.2. I denne tabellen er også problemstørrelsen med antall variable og restriksjoner presentert. For de mindre datasettene finner XpressMP en optimal løsning. For hele problemet for materielltype 69-2 har man funnet en optimal løsning. Dette er et stort problem i antall dagsløp, men lite når det gjelder antall vedlikehold, og hvor godt kilometerintervallene må utnyttes for å finne en optimal løsning. Løsningstidene for typer5, typer7 og typer8 er bedre enn de som ble testet og funnet i Bjorvand and Risberg [2004][4]. Noe av dette skyldes at det har blitt generert en bedre nedre grense for problemet i denne oppgaven.

Man ser at for de store datasettene blir det ingen optimal løsning. Hvis man kjører modellene tilstrekkelig lenge nok, vil de kunne returnere en lovlig løsning med omtrent like mange vedlikehold som NSB har i planen i dag. For de mindre datasettene genererer modellen materiellturneringsplaner med like mange eller færre vedlikehold enn NSB har i planen i dag.

[4]Disse resultatene er ikke helt sammenliknbare siden de er kjørt på ulike maskiner. I Bjorvand and Risberg [2004] var det en Celeron 2,4GHz prosessor og 256 MB minne.

5.2.3 Benders algoritme

I tabell 5.3 er størrelsen for de ulike problemene i Benders algoritme vist. For størrelsen til masterproblemet er ikke lovlighetskutt inkludert.

Tabell 5.3: Problemstørrelse for datasettene med Benders algoritme.

Datasett	**Master**		**Fase1**		**Sub**	
	Var.	Restr.	Var.	Restr.	Var.	Restr.
typer1	106/75	60/39	131/94	117/80	14/10	117/9
typer2	248/246	88/85	301/238	273/210	28/16	216/13
typer3	191/189	88/85	244/187	216/159	28/16	216/13
typer4	242/240	88/85	295/237	267/209	28/16	267/13
typer5[a]	665/663	257/212	830/464	746/422	84/26	746/22
typer5	717/715	257/212	882/516	798/474	84/26	798/22
typer6	308/306	197/143	417/241	361/213	56/20	361/18
typer7	457/455	116/114	538/443	496/401	42/26	496/22
typer8	713/711	144/142	822/700	766/644	56[b]	766[b]
typer72	3026/3024	298/296	3233/2528	3128/2423	105[b]	3128[b]
typer692	3943/3941	322/320	4130/2987	4035/2892	95/87	4035/85
typer693	22781/22779	784/782	23430/19924	23104/19598	326[b]	23104[b]
typer$_{\text{ALLE}}$	29747/29745	2366/1290	32900/25447	31322/24921	1578[b]	31322[b]

a Tomtog = 0, kun lov med tomtog til og fra vedlikehold

b Uløselig, har ikke funnet en lovlig løsning, og det er derfor ikke utført presolve.

Benders masterproblem har omtrent lik løsningstid om man lineærrelakserer binærkravet eller ikke. Noen av variablene får fraksjonelle verdier når problemet lineærrelakseres. Det betyr at masterproblemet ikke har heltallsegenskap. Den innebygde løsningsmetoden i Xpress klarer å løse disse problemene omtrent like fort. Dette ble testet ved å løse problemet med datasett typer 72 med lineærrelaksering og ikke for et bestemt antall iterasjoner, se tabell 5.4.

Tabell 5.4: Sammenlikning mellom LP masterproblem og heltalls-masterproblem

	LPrelaksert	Heltall
Datasett	typer72	typer72
Lovlighetskutt	25	25
Tid Master	160.4s	163.8s
Tid Fase 1	82.2s	85.6
Totaltid	242.6s	249.4s

I tabell 5.5 og tabell 5.6 er resultat etter testing med ulike datasett for Benders algoritme og Benders relakserte algoritme presentert. Hvis man ser på resultatene med å løse problemene med Benders algoritme der binærkravet først er lineærrelaksert ser man at

Tabell 5.5: Resultat etter kjøring av Benders algoritme uten lineærrelaksering.

Inndata	Uke[d]	Master	Fase 1		Sub	Total tid	Kutt	TT	VLH
			Løse	Skrive[c]					
typer1	1	0.3s	0.05s	0.09s	0.02s	0.4s	2	325.1	1
typer2	1	0.2s	0.06s	0.05s	0.08s	0.4s	1	768.4	3
typer3[a]	1	52.8s	1.9s	24.9s	0.08s	79.7s	90	6	3
typer4	1	0.265s	0.1s	0.1s	0.1s	0.6s	2	478.4	3
typer5[a]	1	550.4s	11.0s	206.7s	0.2s	768.3s	146	278.8	4
typer5[b]	1	875.7s	15.3s	322.4s	0.2s	1213.6s	167	245.6	4
typer6	1	0.6s	0.1s	0.2s	0.1s	1.0s	3	625.1	2
typer7	1	23.8s	2.1s	10.2s	0.1s	36.2s	33	710.5	4
typer8	1	37618.4s	104.8s	13556.3s	-	51279.5s**	960	-	-
typer72	1	126367s	445s	43442s	-	170254s**	768	-	-
typer692	1	6.0s	3.5s	3.4s	0.6s	13.5s	4	1022.1	2
typer693	1	55161.8s	1297.5s	19089.8s	-	75549.1s**	93	-	-
typerALLE	1	53460s	1751.2s	18098.4s	-	73309.6s**	64	-	-

** Modellen ble stoppet før optimal løsning ble funnet.

a TT = 0 (ikke tillatt med tomtog)

b TT = 1 (Tillatt med tomtog mellom alle noder)

c Skrive er tiden det tar å oppdatere kutt i .dat-fil

d L = Like uker, U = Ulike uker

for et av datasettene, typer3, sparer man mange kutt, og det blir mye kortere løsningstid. Hvis man ser på de andre datasettene i tabell 5.6 er løsningstiden blitt lengre med den relakserte versjonen av algoritmen. Det virker tilfeldig hvilken av disse to algoritmene som vil gi det beste svaret, for det kommer helt an på datasettet. Siden de to master-modellene genererer ulike løsninger, vil det bli ulike lovlighetskutt, og løsningstiden vil være avhengig av hvor bra disse kuttene avskjærer løsningsrommet. Det vil si at i noen tilfeller vil det være best med den ene algoritmen og i andre tilfeller vil det være best med den andre algoritmen. Men argumentet med at det vil bli generert flere kutt når masterproblemet er enklere å løse holder ikke i dette tilfellet siden problemene er like lette å løse i XpressMP. Derfor har jeg valgt å fokusere på den algoritmen som ikke er relaksert, siden den er raskest for de fleste datasettene, og kuttene blir nærmere konvekse hull siden man løser et heltallsproblem.

For datasett typer5 bruker algoritmen lang tid på å finne en løsning. Dette er et flertyperproblem, og det har mange vedlikeholdsmuligheter i forhold til antall dagsløp. Der man har sagt at det ikke er lov til å kjøre tomtog, for typer5 og typer3, bruker algoritmen også lang tid på å finne løsninger.

Kjøretiden til masterproblemet er økende med antall lovlighetskutt. For typer8 løste jeg tre masterproblemer, det første med ett lovlighetskutt og det andre med 111 lovlighetskutt. Kjøretiden for det første problemet var 0,2 sekunder, mens for det andre var det 6,1 sekunder. Tida som ble brukt til å løse heltallsproblemet i XpressMP var 0,1 sekund for det første problemet og 0,3 sekunder for det andre problemet. Når antall

lovlighetskutt var 1524 var kjøretiden for problemet 100 sekunder, mens løsetiden for heltallsproblemet i XpressMP er 2,0 sekunder. Dette resultatet[5], sammen med skrivetiden fra fase1-problemet, viser at algoritmen er programmert veldig ueffektivt. Det går med mye tid på å tegne opp problemene og skrive data til fil. Tiden som blir brukt til å løse selve problemet er ikke så avhengig av antall lovlighetskutt.

Tabell 5.6: Resultat etter kjøring av Benders algoritme med lineærrelaksering.

Inndata	Uke	Master	Fase 1		Sub	Total tid	Kutt	TT	VLH
			Løse	Skrive[d]					
typer2	1	14.5s	1.2s	6.9s	0.03s	22.6s	39	768.4	3
typer3[a]	1	10.2s	0.6s	4.0s	0.02s	14.8s	38	6	3
typer4	1	81.3s	2.8s	34.5s	0.02s	118.6s	100	478.4	3
typer6	1	0.5s	0.1s	0.2s	0.1s	0.9s	3	625.1	2
typer692	1	80.7s	11.0s	30.5s	0.3s	122.5s	16	1022.1	2

a TT = 0 (ikke tillatt med tomtog)

b TT = 1 (Tillatt med tomtog mellom alle noder)

c Skrive er tiden det tar å oppdatere kutt i .dat-fil

5.2.4 Sammenlikning mellom LØS1 og Benders algoritme

Hvis man sammenlikner de to løsningsmetodene fra tabell 5.2 og tabell 5.5 så ser man at algoritmen løser de fleste av de enkle problemene raskere enn LØS1. Når Benders algoritme må kutte mange ganger for å få en lovlig løsning, øker løsningstiden i forhold til LØS1. Dette kan i hovedsak skyldes at problemene må lastes inn og tegnes opp på nytt mellom hver iterasjon. Denne ekstratiden har ikke LØS1 siden hele problemet kun lastes inn en gang. I tillegg kan det skyldes at Benders algoritme konvergerer sakte.

For datasett typer692 løser algoritmen problemet mye raskere enn LØS1. Dette skyldes blant annet at algorimen har laget en bedre nedre grense for problemet enn det formuleringen i Xpress har. Ved å sette den nedre grensen til typer692 i LØS1 til 2 i stedet for til 1, ble det mye kortere løsningstid i Xpress, men fremdeles er algoritmen betraktelig raskere.

Det er et problem som ikke Benders algoritme klarer å løse, men som Xpress klarer, og dette er datasett typer8. Datasett typer3, typer5 og typer7 er Xpress raskere til å løse enn algoritmen, mens algoritmen er raskere til å løse typer1, typer2, typer4, typer6 og typer692.

LØS1 kan gi lovlige løsninger for de fleste problemene bare de får kjørt tilstrekkelig lenge, men man vet ikke hvor langt unna optimalitet disse løsningene er, annet enn at man kan sammenlikne med den nedre grensen for problemet. Problemet med Benders algoritme er at man ikke har en lovlig løsning før man har den optimale løsningen.

Ut i fra de relativt få testsettene som modellene er testet på kan jeg ikke trekke en entydig konklusjon på hvilken løsningsmetode som er best fordi for noen datasett er

[5]Denne testen er utført på en maskin med 2,4GHz prosessor og 256MB minne. Dette resultatet viser at selv med en nyere harddisk brukes det mye tid på å laste inn fra datafiler.

formuleringen i Xpress best, og for noen andre er Benders algoritme best. Hvert problem i Benders algoritme er mye mindre å løse enn hele problemet i Xpress. Kjøretiden per masterproblem med Benders algoritme er økende med antall lovlighetskutt. Dette skyldes i større grad at problemene blir større å laste inn enn at problemet blir verre å løse. For typer692 trenger ikke algoritmen så mange lovlighetskutt før den finner en lovlig løsning, og da er løsningstiden mye bedre enn for LØS1. Det samme gjelder for de mindre datasettene. Men LØS1 har en fordel fordi LØS1 finner optimal løsning for alle datasettene som algoritmen også klarer å løse til optimalitet, selv om den er tregere for noen av problemene, finner den løsning på flere problemer enn det algoritmen gjør.

Kapittel 6

Diskusjon og videre arbeid

I dette kapitlet vurderes nytteverdien til optimeringsmodellen som er utviklet i dette arbeidet opp mot de kvalitetskriterier som stilles til en modell som lager materiellturneringsplaner. Modellens kvalitet vurderes ved dens kvalitative og kvantitative egenskaper.

I dette kapitlet ønsker jeg å svare på spørsmålene:

- Kan vedlikehold integreres i en optimeringsmodell, og er det noe å spare på å planlegge vedlikehold med en operasjonsanalytisk modell?
- Egner modellen seg til å planlegge vedlikehold på det taktiske nivået?
- Er løsningstiden for den operasjonsanalytiske modellen akseptabel for bruksområdet modellen er tiltenkt?

6.1 Beslutningsstøtteverktøy

Den utviklede optimeringsmodellen er ikke tiltenkt som en egen frittstående applikasjon. Siden dagsløp er inndata til modellen, kreves det at disse er generert i forkant. SINTEF samarbeider med NSB for å lage et planleggingsverktøy for materiellturnering, og dette verktøyet kan kanskje i fremtiden lage dagsløp som kan være inndata til en slik vedlikeholdsmodell. Jeg har brukt dagsløp som er planlagt manuelt som inndata.

Det er ikke prioritert å lage et brukervennlig grensesnitt til modellen. Hvis modellen skal kunne brukes i NSB må dette lages. Det er nødvendig at brukergrensesnittet til denne modellen samsvarer med grensesnittet til de andre modellene som den er tiltenkt å brukes sammen med, for at den skal ha en god nytteverdi i NSB.

Det må i tillegg gjøres noe med formatet på inndata og utdata til modellen. Hvis flere modeller skal kommunisere med hverandre i et planleggingsverktøy er det en fordel at de leser og skriver data til samme database. Det krever da blant annet at datamaterialet er på samme format, og at man prøver å unngå dobbeltlagring i så stor grad som mulig.

For materiellplanlegging har NSB et datasystem som heter *Resource Plan* der materiellplanene blir registrert i dag. Dette datasystemet foreslår ikke løsninger, men kan brukes til feildeteksjon og visualisering av planer. For at optimeringsmodellen skal kunne brukes i et fremtidig beslutningsstøttesystem bør planene som blir generert av beslutningsstøttesystemet være på et format slik at planene er kompatible med *Resource Plan*.

Med dette menes ikke at beslutningsstøttesystemet skal lage endelige planer som skal registreres direkte i *Resource Plan*, men at materiellplanleggeren skal få en enklere jobb med å registrere planer når han først har generert en plan og sørget for at alle kravene blir tatt hensyn til.

6.2 Kvalitativ vurdering av modell

De kvalitative egenskapene til en optimeringsmodell vurderes ved hvor godt modellen støtter og implementerer kravene i den aktuelle planleggingsfasen. Kvaliteten til modellen er derfor avhengig av detaljnivået til modellen i forhold til detaljnivået i planleggingsfasen.

En kvalitetsvurdering av vedlikeholdsmodellen vil i tillegg inneholde en vurdering av hvor god kilometerutnyttelse den har i forhold til det en materiellplanlegger kan klare å planlegge manuelt ut over ren tidsbesparelse.

6.2.1 Planleggingsfase

I utgangspunktet var det tenkt at optimeringsmodellen skulle være en modell for vedlikeholdsplanlegging på det taktiske nivået (årlig plan). I materiellplanleggingsprosessen i dag gjøres alt manuelt, og optimeringsmodellen som er utviklet vil ikke kunne generere en ferdig årlig plan. Til det mangler flere av kravene som stilles til en materiellturneringsplan. Optimeringsmodellen betrakter blant annet ikke hensettingskapasitet som er et av kravene som stilles til en materiellturneringsplan på det taktiske nivået. Tomtogsbehov blir identifisert, men løsningen modellen gir sier ikke noe om når tomtogene skal kjøres.

Valget av når tomtogene skal kjøre er valg som planleggeren vil kunne gjøre mer fornuftig enn denne modellen vil klare. Hvis modellen skal lage en like god plan som planleggeren, kreves det mye mer av inndataene som tidligere nevnt i kapittel 4.1. Derfor vil modellen egne seg bra for en tidlig taktisk fase, ved å generere planer som tar hensyn til vedlikeholdskrav, men etter at man har fått inn vedlikehold i planen må materiellplanleggeren selv sørge for at den ferdige planen tilfredsstiller alle de andre kravene til en materiellturneringsplan.

Modellen vil kunne gi en bedre nedre grense på antall vedlikehold i planen enn den nedre grensen som blir beregnet i dag. Den nedre grensen på antall vedlikehold i dag beregnes ut i fra den totale kjørelengden delt på kilometergrensen. Det vil si at modellen kan gi en bedre praktisk optimal grense for antall vedlikehold.

I den strategiske planleggingsfasen kreves ikke et så stort detaljnivå på planene som for årlig plan. I strategisk planlegging jobber man gjerne med å utrede nye konseptuelle produksjonsmodeller og vurdere konsekvensene av disse. For en plan på dette nivået er det ikke interessant å vite om det blir stående et tog for mye på en stasjon på en natt, eller om planen bruker ti vedlikehold i stedet for ni vedlikehold. Det er ikke det som gjør at den ene konseptuelle modellen foretrekkes framfor den andre. På dette nivået er det heller for eksempel kundebehov og kjørefrekvens som avgjør valget. På det strategiske nivået kan det være ønskelig å detaljere noen av produksjonsmodellene for å se om de er gjennomførbare. Da kan denne optimeringsmodellen brukes.

Siden modellen inkluderer vedlikeholdskrav, kan man i den strategiske fasen begynne å innlede forhandlinger med vedlikeholdsenheten siden man da har et anslag på volumbehovet for vedlikeholdet.

Modellen kan altså brukes til:

- Få en bedre praktisk nedre grense for vedlikeholdsbehovet.
- Koble sammen dagsløp og vedlikehold til et utkast til en materiellturneringsplan som materiellplanleggeren kan jobbe videre med for å lage en endelig plan. For dette utkastet vil tomtogsbehovet bli identifisert, men hensettingskapasitet tas ikke hensyn til.
- Gjøre konsekvensanalyser. Man kan endre parametre som for eksempel lengde på vedlikehold, kapasitet på vedlikeholdsbaser, kilometergrensen for de forskjellige materielltypene, mulige vedlikeholdsbaser og endre hvilke materielltyper som kan vedlikeholdes på de ulike vedlikeholdsbasene for å se hva slags konsekvenser dette har på materiellturneringen. Man kan også sette ulike kostnader strekningene man kjører tomtog og på vedlikeholdsmulighetene for å se konsekvensene.
- Planlegger får på et tidligere tidspunkt et mer detaljert uttrykk av konseptuelle modeller, og det fører til at man kan arbeide med flere alternativer samtidig.
- Sjekke gjennomførbarheten til en konseptuell produksjonsmodell når man begynner å detaljere de konseptuelle modellene på strategisk nivå. Man får da et godt anslag i en tidlig fase på vedlikeholdsbehov, tomtogsbehov og materiellbehov, og ved at man finner ut dette tidlig i fasen kan man starte forhandlinger med vedlikeholderen, de som planlegger personell i forhold til tomtogskjøring og de som bestemmer over materiellet.

Det største problemet til optimeringsmodellen er at den har lang løsningstid for de større problemene som vist i kapittel 5, og man har heller ingen garanti for at modellen vil finne en god lovlig løsningen innenfor en rimelig tid[1]. For problemet til hele materielltype 69-3 finner man en løsning som bruker et vedlikehold mer enn NSBs plan etter ca. 20 timer. For hele lokaltogsproblemet finner man en løsning som bruker 6 vedlikehold mer enn NSBs plan etter ca. 20 timer.

Når man skal lage en årlig plan kan man ikke vente lenge på at planleggingsverktøyet skal lage et utkast på en plan, siden det ikke er en ferdig plan som kommer ut. Planen må bearbeides ved at planleggeren må ta hensyn til de kravene som modellen ikke inkluderer. Hvis planleggeren ønsker å endre på noen av parametrene til modellen for å generere et annet planutkast, slik at han er et annet utgangspunkt til å generere en ferdig plan, vil lang løsningstid for optimeringsmodellen føre til at denne prosessen tar lang tid.

For den strategiske planleggingsfasen vil optimeringsmodellen egne seg bedre siden man har enda bedre tid i den strategiske fasen enn i den taktiske fasen til å vente på at modellen skal generere en plan, samtidig som modellen har med de kravene som stilles til en materiellplan i denne planleggingsfasen.

[1]Når problemet blir løst med algoritmen finner man ingen løsning for de største datasettene.

6.2.2 Kilometerutnyttelse

NSBs taktiske plan utnytter kilometerintervallene ca. 80 %, det vil si at et materiell-individ i gjennomsnitt vedlikeholdes hver 8.000 km når den øvre grensen er 10.000km. Kjøringen gjennomføres med ca. 70 - 80% utnyttelse.

Hvor godt man kan utnytte kilometerintervallene er avhengig av dagsløpenes lengde og lengden på kilometerintervallene. Ved å lage en optimeringsbasert vedlikeholdsmodell er det interessant å se hvor nært opptil 100% utnyttelse det er mulig å planlegge, samtidig er det interessant å sammenlikne planen fra modellen med en manuelt generert plan. For å kunne trekke en konklusjon på dette, trenger man optimale løsninger av litt større problemer enn det som er funnet i dette arbeidet. Datasett typer692 er et stort problem i den forstand at det er mange dagsløp og mulige vedlikehold, men et lite problem når det gjelder antall dagsløpskilometer og vedlikeholdsbehov. For dette datasettet er kilometerutnyttelsen kun 50% siden man akkurat må ha to vedlikehold i stedet for ett for å generere en optimal plan.

Modellen har ikke brukt flere vedlikehold enn den totale kjørelengden tilsier at man skal bruke for de datasettene man har funnet optimal løsning. For datasettene der man kun har funnet lovlige løsninger i Xpress og ingen løsning med Benders algoritme, varierer kilometerutnyttelsen for løsningene i forhold til NSB noe. For datasett for hele materielltype 72 (typer72) finner modellen en løsning med bedre kilometerutnyttelse enn NSB har i dag. For datasettet med hele problemet (typerALLE) samt datasettet for hele materielltype 69-3 (typer693) finner modellen en løsning med dårligere kilometerutnyttelse enn det NSBs taktiske plan har i dag.

6.3 Kvantitativ vurdering av modell

Den kvantitative vurderingen av modellen gjøres ved hjelp av forhåndsgenererte testsett for å studere hva slags planer som modellen genererer og løsningstidene.

6.3.1 Kritikk av testsett

Det er laget testsett med ulikt antall dagsløps- og vedlikeholdsnoder, og disse nodene er reelle dagsløp og vedlikehold som NSB bruker i dag. Når testsettene har blitt generert er det mange av testsettene som kun er utvidelser fra de mindre testsettene, det vil si at de samme nodene er med i flere sett, bare at i de større settene er flere dagsløpsnoder og vedlikeholdsnoder inkludert. Ulempen ved å gjøre det på denne måten, er at det kan være noen av dagsløpskoblingene som kan være verre/enklere å få til enn andre. Og for mindre testsett kan dette få store utslag. Testsettene er også i hovedsak knyttet til en pendel. Alle settene som inneholder materielltype 72 har dagsløp fra en pendel. Det er kun datasett typer72 som også har dagsløp fra en annen pendel for materielltype 72. For materielltype 69-3 har dagsløp blitt tilfeldig utvalgt. Kun for datasettene typer692 og typer693 er det dagsløp fra alle pendlene.

Siden modellen i hovedsak er testet på et begrenset antall pendler og med lite variasjoner, kan man ikke trekke en endelig konklusjon på hvor godt modellen egner seg for vedlikeholdsplanlegging for lokaltog Østlandet i NSB. Siden det er en begrenset mengde

med testsett kan man heller ikke trekke endelige konklusjoner på hva som gjør at noen av datasettene får lang løsningstid, mens andre løses enklere.

6.3.2 Løsningstid

Ut i fra resultatene etter testingen ser det ut som datasett som krever flere vedlikehold er verre å løse enn datasett som krever få vedlikehold. Problemene som har få vedlikehold samtidig som det ikke kreves stor kilometerutnyttelsen, er enklest å løse. Datasett typer 692 løses fort med algoritmen, og relativt raskt med LØS1. I dette datasettet er det mange noder, men liten kjørelengde og dermed få vedlikehold.

Samtidig ser man også at å generere en "riktig" nedre grense er viktig. For LØS1 ble det ikke funnet en optimal løsning for typer692 når den nedre grensen var ett vedlikehold. Ved nedre grense på to vedlikehold fant algoritmen og LØS1 en optimal løsning etter kortere tid.

For de fleste av de mindre problemene er algoritmen best til å løse problemet. For de større datasettene varierer det mer hvilken av metodene som er best. Algoritmen er best for typer 692, mens LØS1 er best for typer7 og typer8. For datasettene typer72, typer693 og typerALLE får man ikke optimal løsning ved noen av løsningsmetodene. LØS1 vil være best for disse testsettene siden LØS1 finner lovlige løsninger.

6.3.3 Vedlikehold

Modellen som er formulert beskriver vedlikeholdet og vedlikeholdskravene til NSB slik som det er i virkeligheten. Unntaket er at optimeringsmodellen er mye strengere på at man må ferdigstille et vedlikehold før man begynner på det neste. Hvis modellen skulle tillatt at man kunne bytte mellom å vedlikeholde flere tog, ville det ført til mange flere beslutningsvariable, og litt mer avanserte vedlikeholdsrestriksjoner. I tillegg ville det blitt en mer komplisert restriksjon for å sørge for at kapasiteten på vedlikeholdsbasene ble overholdt.

Selv om vedlikeholdsutførelsen er litt mer fleksibel i virkeligheten, er det vist i dette arbeidet at man kan spare inn på antall vedlikehold i materiellturneringsplanen ved å bruke en optimeringsmodell framfor manuell planlegging. For datasett typer8 og typer5, løst med algoritmen og LØS1, lager modellen en plan med færre vedlikehold enn planen som NSB har i dag. Også for datasett typer72 finner LØS1 en lovlig løsning som har færre vedlikehold enn NSBs plan.

For de datasettene man kan løse til optimalitet med LØS1 og Benders algoritme, bruker man like mange eller færre vedlikehold enn den taktiske planen til NSB.

6.3.4 Tomtogskilometer

Mange av avstandene mellom stasjoner i datasettene er kun anslag og ikke den faktiske avstanden, så jeg kan ikke sammenlikne tomtogskilometrene i planen til NSB i dag med tomtogskilometrene som modellen gir, og si at den ene vil være bedre enn den andre på bakgrunn av det. Derfor er det ingen kommentarer og sammenlikninger på tomtogsbehovet i planene. Siden hensettingskapasitet er utelatt fra modellen, vil tomtogsbehovet fra modellen sannsynligvis være bedre enn det i virkeligheten er. Noen av togsettene

hensettes sannsynligvis på en stasjon som ikke er en direkte mellomstasjon mellom en endestasjon og en startstasjon når hensettingskapasitet er inkludert i modellen, og derfor øker tomtogsbehovet.

6.4 Løsningstider

Uansett om modellen ønskes brukt i taktisk eller strategisk planlegging i NSB, er det en fordel med raskere løsningstid for å kunne ta modellen i bruk. I dette delkapitlet vil det presenteres forslag til å få forbedret løsningstidene for problemet.

For LØS1 har jeg ikke kommet med noen spesielle forbedringsforslag for å redusere løsningstiden, siden XpressMP har en innebygget løsningsalgoritme, og denne er optimalisert av Dash Optimization [www.dashoptimization.com]. Et generelt råd er å sjekke om restriksjonene kan formuleres strammere, og dette er også viktig for modellene i Benders algoritme.

6.4.1 Sensitivitetsanalyser

Før man begynner å se på metoder for å redusere løsningstidene til et problem, kan det være smart å prøve å finne årsaker til hva som gjør at løsningstiden er lengre for enkelte problem i forhold til andre. For å finne ut dette bør man utføre sensitivitetsanalyser for å se for eksempel hva en endring av kilometergrensen har å si for løsningstiden for problemene. Hvis man endrer kapasiteten på vedlikeholdsbasen slik at det blir flere mulige vedlikehold, vil dette også sannsynligvis gjøre noe med løsningstiden. Andre parametre/inndata man kan endre på er lengde på dagsløp, sette ulik kostnad på vedlikeholdene og på strekningene for tomtogskjøring. Man bør også utføre en analyse på kostnadsfunksjonen i modellen. Det er ikke prøvd ut ulike forhold mellom kostnadene for vedlikehold og tomtogskilometer, og dette vil sannsynligvis påvirke løsningstiden for modellen.

6.4.2 Løsningstid for Benders algoritme

Jeg har funnet ut at en mer effektiv implementasjon, bedre initielle kutt og omformulering av restriksjoner er noe som kan redusere løsningstiden for Benders algoritme sammen med kuttoptimalisering og nettverksalgoritmer. For de som har mer erfaring med Benders algoritme vil man sikkert vite om flere metoder for å få algoritmen til å konvergere raskere enn det som vil bli kommentert her.

Mer effektiv implementasjon

I dette arbeidet har jeg lagt mest vekt på å finne måter for å få Benders algoritme til å løse problemet med færrest mulig iterasjoner og metoder for å få masterproblemet til å løses raskere. For å få en mer effektiv løsningsprosedyre, bør man se om man kan effektivisere lesing og skriving til datafilene. Det er sannsynligvis mulig å få den implementerte koden for hver av modellen i Benders algoritme i XpressMP mer effektiv, men samtidig blir man ikke kvitt det problemet at problemene må lastes inn på nytt for hver iterasjon. For å få den mest effektive løsningsprosedyren, bør man derfor implementere

algoritmen i et programmeringsspråk. I Cordeau et al. [2001b] ble Benders algoritme implementert i C, og CPLEX Callable Library ble brukt for å løse de lineære problemene og heltallsproblemene til algoritmen.

Ved å se på problemet og problemstørrelsene som blir løst i Cordeau et al. [2001b][2] ser det ut som om det vil være en stor fordel å implementere algoritmen i et eget programmeringsspråk. Ut i fra erfaringer fra Aschehoug and Fodstad [2002] bør man velge c++ framfor java som programmeringsspråk når man benytter XpressMP som optimeringsverktøy.

Initielle kutt

Man må se om det er mulig å si noe mer om løsningen eller det lovlige problemområdet før man begynner å kjøre algoritmen. Er det for eksempel mulig å generere fasetter for å definere deler av lovlighetsområdet før man kjører algoritmen? I Ziarati et al. [1999] utledes fasetter for deres problem som de løser med Branch & Cut og Dantzig-Wolfe dekomponering, samtidig som de også ser på "cutting plane", det vil si måter å formulere restriksjonene nærmere heltallsløsningene.

I testingen ble det vist at jo mer man kunne si om den nedre grensen til problemet, jo bedre løsningstid ble det. For eksempel kan man se forskjell i løsningstid for typer692 for LØS1 i tabell 5.2 der den nedre grensen er ett vedlikehold og to vedlikehold. Tilsvarende forbedring i løsningstid finner man også med Benders algoritme.

Omformulere restriksjoner i masterproblemet

I Magnanti and Wong [1981] blir det kommentert at hvordan modellene og noen av restriksjonene blir formulert også kan ha innvirkning på løsningstiden til Benders algoritme. I denne oppgaven har jeg ikke testet ulike formuleringer på restriksjonene.

Jeg har brukt CG-ulikheter for å formulere lovlighetskuttene, men man kan kanskje finne andre måter å formulere lovlighetskuttene på som vil fungere bedre for dette problemet.

I Alfieri et al. [2003] genereres lovlige ulikheter, såkalte valid inequalities, for noen av restriksjonene, der målet er kortere løsningstid fordi restriksjonene blir formulert nærmere konvekse hull. Det kan ses videre på om dette vil være en mulighet for Benders masterproblem også, om restriksjonene eller lovlighetskuttene kan formuleres enda nærmere konvekse hull. For eksempel ser Escudero and Muños [1998] på hvordan man kan formulere strammere restriksjoner for binære programmer.

Kuttoptimalisering

For de fleste av de mindre datasettene er Benders algoritme raskere til å løse problemene enn det LØS1 er. Problemene blir verre å løse når mengden med kutt i masterproblemet øker, og det kan ses nærmere på om det er mulig å optimalisere mengden med kutt i

[2]Modellen som blir løst i Cordeau et al. [2001b] er ikke lik den som blir utviklet i denne oppgaven, men de klarer å løse mye større problemer enn det som blir gjort i dette arbeidet. I tillegg til å lineærrelaksere masterproblemet, relakseres også noen restriksjoner for å kunne legge til lovlighetskutt når problemet er enklere å løse. Dette førte til mye forbedring i løsningstid for algoritmen.

masterproblemet. I denne oppgaven ble det fokusert på å fjerne kutt som ble dominert av andre kutt, og det var omtrent ingen kutt som kunne fjernes. Dette kan skyldes at kriteriet for å kunne si at et kutt ble dominert var unødvendig strengt. Det er prøvd å slakke litt på kravet ved å kun kreve at den ene ulikheten i sjekken for det dominerende problemet blir oppfylt, men det førte heller ikke til færre aktive kutt. Man bør se om man kan finne redundante kutt i mengden med lovlighetskutt på en annen måte. Hvis man programmerer algoritmen i et programmeringsspråk vil sannsynligvis ikke denne kuttmengden ha så stor betydning, siden det i hovedsak det å lese inn og ut fra datafilene som tar tid, og ikke det å løse selve masterproblemet.

Nettverksalgoritmer

Masterproblemet er et nettverksproblem, så man kan se nærmere på algoritmer som kan løse dette problemet raskere enn XpressMP kan. Problemet har ikke heltallsegenskap, så det må implementeres en heltallsalgoritme.

6.4.3 Redusere kompleksiteten

I en ideell verden skal planleggingsverktøyet støtte planleggingsprosessen, men i noen tilfeller kan det være nødvendig at man tilpasser prosessen til verktøyet fordi verktøyet har noen begrensninger. For vedlikeholdsproblemet for lokaltog Østlandet er det minst to måter å redusere problemstørrelsen på.

Lengre dagsløp

For å få enklere problem å løse kan man prøve å redusere antallet dagsløp i modellen ved å generere noen koblinger der flere dagsløp allerede er koblet sammen. Ved noen stasjoner er kravet til rekkefølge på togsettene viktig, og der balansen enkelt går opp, kan man si at det som kommer sist inn skal først ut igjen, og bestemme at de to dagsløpene skal knyttes sammen. Da vil man få en mengde med lengre dagsløp i modellen. For å få til lengre koblinger i modellen må man endre litt på den implementerte modellen. I stedet for at dagsløpet kun er gitt med dagen det starter på, må også sluttdag inkluderes. Fordelen med å lage lengre dagsløp er at man får tatt bedre hensyn til kravet om rekkefølger på stasjonen, samtidig som antallet variabler i modellen blir mindre. Ulempen er at det kan bli vanskeligere å utnytte kilometerintervallet i optimeringen, siden alle lengdene blir lengre. I tillegg er det jo ikke sikkert at man får den optimale løsningen ved å forhåndsgenerere noen koblinger. Denne forhåndsgenereringen av koblinger kan gjøres ved hjelp av en algoritme (som ikke må være basert på optimering). Eventuelt kan materiellplanleggeren selv legge inn slike koblinger ettersom han finner det fornuftig med sin erfaring.

Løse et materiellturneringsproblem per pendel

Problemstørrelsene som har blitt løst til optimalitet kan sammenliknes med problemstørrelsen for problemet for en eller flere av lokaltogspendlene. Det er i alt åtte pendler, og det blir da åtte delproblemer hvis man løser et problem per pendel.

Hvis man ønsker å løse materiellturneringsproblemet per pendel, må dagsløpene være generert per pendel, eventuelt per noen få pendler. Datasett typer8 er alle dagsløpene for materielltype 72 på en av pendlene. I tillegg benyttes type 72 på en av de andre pendlene. Typer 69-3 brukes i hovedsak på de seks andre pendlene. Materiellet til type 69-3 turnerer ikke adskilt per pendel i dag. Typer 69-2 brukes i hovedsak i rushtiden, enten som påsett (til et annet sett) eller som et ekstratog på flere av pendlene.

Man kan prøve å løse et flertyperproblem per pendel, eller flere entypeproblemer per en mengde med pendler. Siden det er få togsett av noen materielltyper på noen av pendlene, vil det bli nærmere en optimal løsning hvis man slår sammen materiellet på noen av pendelene og løser et entypeproblem over flere pendler.

Problemet med å løse disse subproblemene er at det fører til suboptimale løsninger. Det blir sannsynlig ikke like bra kilometerutnyttelse som ved å løse hele problemet i ett, og man oppnår ikke alle stordriftsfordelene man kan ha når man kan planlegge flere strekninger sammen. Fordelen med å løse et problem per pendel er at man få en mer robust plan og gjennomføring siden et problem på en pendel vil ha liten innvirkning på de andre pendlene.

Problemstørrelsen for de ulike pendlene varier noe. For et par av pendelene er det mange togsett, og ut i fra testresultatene vet jeg ikke om disse problemene kan løses til optimalitet eller løsningstidene. Dette må sjekkes videre ut, og inndatasettene må lages på bakgrunn av de ulike pendlene.

6.4.4 Andre problemer

Lokaltogsproblemet på Østlandet er stort og komplekst. Modellen som er utviklet i dette arbeidet har blitt utviklet med bakgrunn i dette problemet, og siden dette problemet kun har motorvognsett, er det ikke testet hvordan modellen vil takle problemer der man har lokomotiv og vogner. Motorvognsett brukes på flere av regiontogsstrekningene og for de andre lokaltogsproblemene, så det er flere andre problemer også som modellen kan være aktuell for.

Anbudsutsetting

Jernbanestrekningene i Norge har nå blitt åpnet for konkurranse for både lokaltog og regiontog, og NSB må levere inn anbud på lik linje med andre operatører for å få lov til å kjøre tog på de strekningene som blir konkurranseutsatt. I forrige delkapittel ble det foreslått å redusere kompleksiteten ved å løse et problem per strekning, og det likner mye på problemet man møter når man konkurranseutsetter strekninger. For disse strekningene som blir lagt ut på anbud gjelder det at man ikke kan benytte seg av stordriftsfordeler, og materiell og personell skal derfor kun tilordnes til en turnering på denne strekningen. Det fører til mindre materiellplanleggingsproblemer, siden materiellet kun kan brukes på en bestemt strekning.

Den første strekningen som ble konkurranseutsatt var Gjøvikbanen, pendelen Skøyen-Jaren er en del av denne strekningen, og dermed kan ikke materiellet på denne strekningen turnere sammen med annet lokaltogsmateriell. Foreløpig er det ingen plan over at de andre lokaltogsstrekningene skal konkurranseutsettes, men det er to strekninger for regiontog som skal ut på anbud i neste omgang.

Andre lokaltogsproblemer

For å utvikle modellen i denne oppgaven ble det tatt utgangspunkt i lokaltog på Østlandet. Dette er det mest kompliserte av lokaltogsproblemene, siden det er flere vedlikeholdsbaser og flere materielltyper som betjener lokaltogstrafikken, i tillegg til mange pendler (strekninger).

For de andre lokaltogsproblemene er det kun en vedlikeholdsbase og i hovedsak en materielltype, og det er kun en strekning[3]. Problemstørrelsene er vist i tabell 6.1. Man kan ikke si at modellen vil virke bra for disse problemene uten å teste modellen ordentlig med datasett fra disse strekningene. Men ut i fra problemstørrelsen og de problemene som er løst for lokaltog på Østlandet, ser det ut som om det er mulig å bruke optimeringsmodellen for å løse disse problemene.

Tabell 6.1: Problemstørrelse for lokaltogsproblemene til NSB

By	**Materielltyper**
Bergen	2 sett type 69-2 og 4 sett type 69-2
Oslo	10 sett type 69-2, 43 sett type 69-3 og 15 sett type 72
Stavanger	7 sett type 72
Trondheim*	10 sett type 92

* I tillegg kjører to sett mellom Østersund-Trondheim som turnerer sammen med lokaltogene.

Når optimeringsmodellen har blitt utviklet, har bare materiellplanleggere for lokaltogene i Oslo blitt konfrontert, og jeg vet derfor ikke om de andre lokaltogsproblemene har de samme kravene, eller om prioriteringen på målene er lik. Så det er ikke sikkert at det er nok å kun endre på inndatasettene for å kunne lage materiellturneringsplaner for disse problemene.

6.4.5 Annen løsningalgoritme

Tidligere i oppgaven viste jeg til Lagrangerelaksering og en modellformulering basert på denne relakseringen. Det vil lønne seg å prøve ut denne løsningsalgoritmen hvis man ikke får bedre løsningstid med Benders algoritme for de store datasettene. Siden problemet ikke har heltallsegenskap, vil man finne en bedre nedre grense for problemet med Lagrangerelaksering enn om man tar en lineærrelaksering av det originale problemet.

6.4.6 Alternativ formulering

I Bjorvand and Risberg [2004] ble det kommentert en alternativ måte å formulere vedlikeholdsproblemet på. Siden jeg er funnet lite litteratur på vedlikeholdsproblemet, se kapittel 3.1, kan jeg ikke ut ifra litteraturen si at den ene problemformuleringen vil være bedre enn en annen. Den alternative formuleringen kan derfor prøves ut hvis man ikke får bedre løsningstid med modellformuleringen i denne oppgaven, og man fremdeles ønsker å løse større problemer.

[3]I Trondheim er det flere strekninger: Trondheim-Østersund, Trondheim-Oppdal og Lerkendal-Steinkjer.

Kapittel 7

Konklusjon

I denne oppgaven ble vedlikeholdsplanlegging inkludert i en optimeringsmodell for turneringer av togmateriell i NSB. Det har blitt fokusert på å løse dette vedlikeholdsproblemet, og det er brukt to løsningsstrategier, en formulering av problemet i XpressMP (LØS1) og Benders algoritme. Benders algoritme ble implementert med XpressMP og batch-kode der hvert delproblem har blitt formulert og løst i XpressMP og det ble laget et script i batchkode som kjørte algoritmen. For de fleste av de mindre datasettene ga Benders algoritme bedre løsningstid enn LØS1. For de største datasettene ga ingen av løsningsstrategiene optimal løsning.

Ut i fra testresultatene hadde LØS1 en stor fordel framfor Benders algoritme siden LØS1 fant lovlige løsninger for alle problemer. Med den implementerte Benders algoritme fant man optimal løsning for små problemer, men ingen løsning for de større problemene. LØS1 fant optimal løsning for alle problemene som Benders algoritme fant optimal løsning for. Benders algoritme konvergerte sakte for de større problemene, og hvert masterproblem fikk lengre kjøretid når det ble flere lovlighetskutt lagt til problemet. Det har blitt sett på metoder for å få raskere konvergens og fjerne kutt fra masterproblemet for å redusere løsningstiden, men dette førte ikke til at algoritmen kunne løse de større problemene.

For problemene som ble løst til optimalitet genererte modellen en materiellturneringsplan som brukte like mange eller færre vedlikehold enn NSBs taktiske plan. Dette viser at det er et potensial for å redusere antall vedlikehold i en plan ved hjelp av optimering.

Optimeringsmodellen som ble utviklet i dette arbeidet kan støtte planlegging både i den taktiske og den strategiske planleggingsfasen. I den taktiske fasen kan optimeringsmodellen benyttes til å finne en nedre grense for antall vedlikehold, samt generere førsteutkast av planer. I den strategiske fasen kan modellen benyttes mer til å gjøre ulike konsekvensanalyser og sjekke gjennomførbarheten til produksjonsmodellene, samt finne ut vedlikeholdsbehovet tidligere, og dermed kunne starte forhandlinger med vedlikeholder på et tidligere tidspunkt. For at optimeringsmodellen skal kunne bli integrert i et planleggingsverktøy kreves det en kortere løsningstid enn det som ble funnet i dette arbeidet, i tillegg til et bedre grensesnitt og en grundigere testing av modellen.

For å få raskere løsningstid ble det i første omgang anbefalt å utføre sensitivitetsanalyser for å finne ut hva som påvirker løsningstiden til problemet. Effektivisere implementasjonen av algoritmen, generere flere initielle kutt og omformulere restriksjoner ble der-

etter nevnt for å få raskere løsningstid. Videre ble det foreslått å se på løsningsalgoritmen Lagrangerelaksering hvis ikke Benders algoritme får raskere løsningstid og til slutt formulere problemet med en alternativ formulering som ble vist i Bjorvand and Risberg [2004] for å prøve å redusere løsningstiden. Det ble diskutert hvordan man kan redusere kompleksiteten til vedlikeholdsproblemet ved å ta inn lengre dagsløp (dagsløp over flere dager) eller løse problemene per lokaltogspendel i stedet for hele problemet i ett. I NSB er det andre materiellplanleggingsproblemer som er mindre i størrelse enn lokaltog Østlandet, og det ble også foreslått at man kan se hvordan algoritmen virker for disse problemene.

Bibliografi

E. Abbink, B. van den Berg, L. Kroon, and M. Salomon. Allocation of Railway Rolling Stock for Passenger Trains, 2003. http://papers.ssrn.com/sol3/cf_dev/AbsByAuth.cfm?per_id=337363(Aksessert 27.sept 2004).

A. Alfieri, R. Groot, L. Kroon, and L. Schrijver. Efficient Circulation of Railway Rolling Stock, 2003. http://papers.ssrn.com/sol3/cf_dev/AbsByAuth.cfm?per_id=337363 (27.sept 2004).

K. Aschehoug and M. Fodstad. Optimeringsmodeller innen materiellturnering for NSB. Diplomoppgave, Norges teknisk- naturvitenskapelige universitet, NTNU, 2002.

M. Banihashemi and A. Haghani. A Model for the Multiple Depot Transit Vehicle Scheduling Problem with Route Time Constraints.

J. Beasley. *Lagrangean Relaxation*, chapter 6, pages 243–303 in C. Reeves(Ed): "Modern Heuristic Techniques for Combinatorial Problems". Blackwell ScientificPublications, Oxford, UK, 1993.

J. F. Bender. Partitioning procedures for solving mixed-variables programming problems. *Numerische Mathematik*, 4:238–252, 1962.

J. R. Birge and F. Louveaux. *Introduction to Stochastic Programming*. Springer, New York, 1997.

J. Bjorvand and M. Risberg. Vedlikeholdplanlegging for NSB. Prosjektoppgave, Norges teknisk- naturvitenskapelige universitet, NTNU, 2004.

J.-F. Cordeau, G. Desaulniers, N. Lingaya, F. Soumis, and J. Desrosiers. Simultaneous locomotive and car assignment at VIA Rail Canada. *Transportation Research, Part B*, 35:767–787, 2001a.

J.-F. Cordeau, F. Soumis, and J. Desrosiers. A Benders Decomposition Approach for the Locomotive and Car Assignment Problem. *Transportation Science*, 34(2):133–149, 2000.

J.-F. Cordeau, F. Soumis, and J. Desrosiers. Simultaneous Assignment of Locomotives and Cars to Passenger Trains. *Operations Research*, 49(4):531–548, 2001b.

J.-F. Cordeau, P. Toth, and D. Vigo. A Survey of Optimization Models for Train Routing and Scheduling. *Transportation Science*, 32(4):380–404, 1998.

T. Erlebach, M. Gantenbein, et al. On the complexity of train assignment problems, 2004. http://citeseer.ist.psu.edu/485202 (Aksessert 20.sept 2004).

L. F. Escudero and S. Muños. On characterizing tighter formulations for 0-1 programs. *European Journal of Operational Research*, 106:172–176, 1998.

M. Florian, G. Bushell, j. Ferland, G. Guèrin, and L. Nastansky. The engine scheduling problem in a railway network. *Infor*, 14(2):121–138, 1976.

M. A. Forbes, J. N. Holt, and A. M. Watts. Exact Solution of Locomotive Scheduling Problems. *The Journal of the Operations Research Society*, 42(10):825–831, 1991.

M. A. Forbes, J. N. Holt, and A. M. Watts. An exact algorithm for multiple depot bus scheduling. *European Journal of Operational Research*, 72:115–124, 1994.

R. Freling, R. M. Lentink, L. G. Kroon, and D. Huisman. Shunting of Passenger Train Units in a Railway Station. Report EI2002-26, Econometric Institute, Erasmus University Rotterdam, 2002.

P. Kall and S. W. Wallace. *Stochastic Programming*. John Wiley & Sons, Chichester, 1994.

A. Kokott and A. Löbel. Lagrangean Relaxations and Subgradient Methods for Multiple-Depot Vehicle Scheduling Problems. Preprint SC 96-22, Konrad-Zuse-Zentrum für Informationstechnik Berlin, 1996.

L. G. Kroon and R. A. Zuidwijk. Mathematical models for planning support. ERIM report series ERS-2003-032-LIS, Rotterdam School of Management, Erasmus University Rotterdam, 2003.

R. M. Lentink, P.-J. Fioole, L. G. Kroon, and C. van't Woudt. Applying operations research techniques to planning of train shunting. report ERS-2003-094-LIS, Erasmus Research Institute of Management (ERIM), Erasmus Universiteit Rotterdam, 2003.

N. Lingaya, J.-F. Cordeau, G. Desaulniers, J. Desrosiers, and F. Soumis. Operational car assignment at VIA Rail Canada. *Transportation Research, Part B*, 36:755–778, 2002.

A. Löbel. Experiments with a Dantzig-Wolfe Decomposition for Multiple-Depot Vehicle Scheduling Problems. Preprint SC 97-16, Konrad-Zuse-Zentrum für Informationstechnik Berlin, 1997a.

A. Löbel. Solving Large-Scale Multiple-Depot Vehicle Scheduling Problems. Preprint SC 97-17, Konrad-Zuse-Zentrum für Informationstechnik Berlin, 1997b.

T. L. Magnanti and R. T. Wong. Accelerating Benders Decomposition: Algorithmic Enhancement and Model Selection Criteria. *Operations Research*, 29(3):464–484, 1981.

NSB Materiell Plan. Virksomhetsplan MT-P - oppgaver og grensesnitt internt og eksternt, 2003.

D. McDaniel and M. Devine. A Modified Benders' Partitioning Algorithm for Mixed Integer Programming. *Management Science*, 24(3):312–319, 1977.

G. L. Nemhauser and L. A. Wolsey. *Integer and Combinatoral Optimization*. John Wiley & Sons, New York, 1988.

M. Nordby. Bedriftspresentasjon på NTNU 21.februar 2005, 2005.

NorskJernbaneklubb. http://www.njk.no/ [01.06.2005].

A. Nõu, J. Desrosiers, and F. Soumis. Weekly Locomotive Scheduling at Swedish State Railways. GERAD G-97-35, Ècole des Hautes Ètudes Commerciales de Montrèal, 1997.

NSB. Diverse dokumenter og informasjon fra nsb.

B. Nygreen. TIØ8: Optimering av transport og prosessproduksjon. Forelesning, Benders algoritme, NTNU, 2004.

N. Olsson, M. Indbryn, and M. Veiseth. Konsekvensanalyse og beslutningsstøtte i jernbaneplanlegging. Rapport ST38F2827, SINTEF Teknologiledelse, 2002.

M. Peeters and L. G. Kroon. Circulation of Railway Rolling Stock: A Branch-and-Price Approach. ERIM report series ERS-2002-110-LIS, Rotterdam School of Management, Erasmus University Rotterdam, 2003.

R. L. Rardin. *Optimization in Operations Research*. Prentice Hall, Inc, New Jersey, 1998.

A. B. Rasmussen. Tildeling af vogne og lokomotiver til passagertog med afstandsafhængige begrænsninger, 2004.

K. Roos. Discrete (and Continuous) Optimizations, chapter 8: Cutting Plane algorithms. Forelesningsnotater, Technische Universiteit Delft, 2004.

A. Schrijver. Minimum Circulation of Railway Stock. *CWI Quarterly*, 6(6):205–217, 1993.

T. Tomasgard. Møte på NTNU med T. Tomasgard fra NSB om prosjekt 4.okt, 2004a.

T. Tomasgard. Presentasjon av NSB Persontog Plan for sommerjobbstudenter 15.juni, 2004b.

S. W. Wallace and R. J.-B. Wets. The facets of the polyhedral set determined by the Gale-Hoffman inequalities. *Mathematical Programming*, 62:215–222, 1993.

L. A. Wolsey. *Integer Programming*. John Wiley & Sons, Inc, New York, 1998.

K. Ziarati, F. Soumis, J. Desrosiers, S. Gelinas, and A. Saintonge. Locomotive assignment with heterogeneous consists at CN North America. *European Journal of Operational Research*, 97:281–292, 1997.

K. Ziarati, F. Suomis, J. Desrosiers, and M. M. Solomon. A Branch-First, Cut-Second Approach for Locomotive Assignment. *Management Science*, 45(8):1156–1168, 1999.

Vedlegg A: Innhold på vedlagt CD

På vedlagt cd ligger:

Modeller. Totalmodellen i XpressMP ligger vedlagt. Det samme gjør de tre versjonene av Benders algoritme.

Resultater. Alle resultater som er kommentert i oppgaven ligger vedlagt. Resultatene ligger vedlagt den modellen de hører sammen med.

Testsett. Disse er vedlagt som .dat-fil. Disse er generert ut i fra reelle dagsløp i NSB.

Oppgaven. Pdf-fil av oppgaven ligger vedlagt, i tillegg til .tex-filene som er brukt for å genere pdf-dokumentet.

Vedlegg B: Algoritmer

Det er tre ulike versjoner av Benders algoritme som er testet i denne oppgaven:

- Kjører algoritmen ved å veksle mellom å løse masterproblemet og fase1-problemet.
- Kjøre den lineærrelakserte versjonen av algoritmen først, før man introduserer heltallskrav og kjører algoritmen på nytt.
- Kjører algoritmen ved å veksle mellom å løse masterproblemet og fase1-problemet. For hver 30. iterasjon kjøres kuttmodellen for å se om man kan fjerne noen kutt fra masterproblemet.

Algoritmene er bygget opp litt forskjellig, og i dette vedlegget presenteres de tre algoritmene med hensyn på hvilke modeller fra tabell B.1 som skal kjøres.

Algoritme B.1 *Algoritme for Benders algoritme uten lineærrelaksert algoritme.*

1. *Løs det lineærrelakserte hovedproblemet NSB_ for å finne en nedre grense for antall vedlikehold.*
   (a) *Hvis problemet er uløselig, gå til trinn 7.*
2. *Løs NSB_bender_master_forst.*
   (a) *hvis problemet er uløselig, gå til trinn 7.*
3. *Løs fase1_forst.*
   (a) *Hvis målfunksjonsverdien til fase1-problemet er lik null, gå til subproblemet i trinn 6.*
4. *Løs NSB_bender_master*
5. *Løs fase1*
   (a) *Hvis målfunksjonsverdien til fase1-problemet er større enn null, gå til trinn 4.*
6. *Løs NSB_bender_sub.*
7. *Terminer algoritme.*

Algoritme B.2 *Algoritme for Benders algoritme med lineærrelaksert algoritme.*

Tabell B.1: Oversikt over alle modellene i algoritmene.

Modell	**Forklaring**
NSB_1	Lineærrelaksert modell av vedlikeholdsproblemet
NSB_bender_master_forst	Masterproblem. Litt forskjell når det er første gangen det løses.
NSB_bender_master_forst_relax	Lineærrelaksert masterproblem. Litt forskjell når det er første gang det løses.
NSB_bender_master	Masterproblem.
NSBbender_master_relax	Lineærrelaksert masterproblem.
NSB_bender_sub	Subproblem.
fase1_forst	Fase1 problemet til subproblemet. Litt forskjell når det er første gang det løses.
fase1_forst_relax	Det lineærrelakserte Fase1-problemet. Litt forskjell når det er første gang det løses
fase1	Fase1-problemet til subproblemet.
fase1_relax	Det lineærrelakserte fase1-problemet til subproblemet.
kutt	Sjekk for dominerende kutt.

1. *Løs det lineærrelakserte hovedproblemet NSB_1 for å finne en nedre grense for antall vedlikehold.*

   (a) *Hvis problemet er uløselig, gå til trinn 11.*

2. *Løs NSB_bender_master_forst_relax.*

   (a) *Hvis problemet er uløselig, gå til trinn 11.*

3. *Løs fase1_forst_relax.*

   (a) *Hvis målfunksjonsverdien til fase1-problemet er lik null, begynn å løs heltalls-algoritmen i trinn 6, siden det ikke er generert lovlighetskutt ennå.*

4. *Løs NSB_bender_master_relax*

5. *Løs fase1_relax*

   (a) *Hvis målfunksjonsverdien til fase1-problemet er større enn null, gå til trinn 4 .*

   (b) *Ellers begynn å løse heltallsalgoritmen i trinn 8.*

6. *Løs NSB_bender_master_forst.*

   (a) *Hvis problemet er uløselig, gå til trinn 11.*

7. *Løs fase1_forst.*

   (a) *Hvis målfunksjonsverdien er lik null, gå til trinn 10.*

8. *Løs NSB_bender_master.*

9. *Løs fase1.*

   (a) *Hvis målfunksjonsverdien er større enn null, gå til trinn 8.*

10. *Løs NSB_bender_sub.*

11. *Terminer algoritme.*

Algoritme B.3 *Algoritme for Benders algoritme uten lineærrelaksert algoritme.*

1. *Sett t = 1. Løs det lineærrelakserte hovedproblemet NSB_1 for å finne en nedre grense for antall vedlikehold.*

   (a) *Hvis problemet er uløselig, gå til trinn 8.*

2. *Løs NSB_bender_master_forst.*

   (a) *hvis problemet er uløselig, gå til trinn 8.*

3. *Løs fase1_forst.*

(a) *Hvis målfunksjonsverdien til fase1-problemet er lik null, gå til subproblemet i trinn 7.*

4. *Sett t = t+1, Løs NSB_bender_master*
5. *Løs fase1*
   (a) *Hvis målfunksjonsverdien til fase1-problemet er større enn null*
      i. *hvis t ¡ 30, gå til trinn 4.*
   (b) *Ellers gå til subproblemet i trinn 7.*
6. *Løs Kuttmodell, sett t = 0, gå deretter til trinn 4.*
7. *Løs NSB_bender_sub.*
8. *Terminer algoritme.*

Vedlegg C: Modeller til Bender

Modellene i dette vedlegget er hentet fra Benders algoritme som ikke er lineærrelaksert først. Alle modellene og algoritmene ligger vedlagt på cd. Hver modell returnerer en exit-kode som batch-scriptet benytter for å bestemme den neste modellen som skal kjøre. Disse exit-kodene varierer med henyn på algoritmene.

C.1 Masterproblem

```
model NSB_bender
uses "mmxprs"
uses "mmsystem"

(!-----------------------------------------------------
Masterproblemet til Bender for vedlikeholdsproblemet

Laget av Marianne Risberg
Masteroppgave NTNU våren 2005
-----------------------------------------------------!)



parameters



!-----------------------------------
!Henter løsning fra subproblemet
!-----------------------------------
LOSNINGSUB = 'losning_subproblem.dat'
LOSNINGSUBLOVLIG = 'losning_subproblemstraale.dat'
NETTVERK = 'generert_graf.dat'
end-parameters

STARTTIME := gettime

declarations


UKE: integer
TOMTOG: integer
LIKEUKER: integer
```

```
TYPE: set of integer !togtyper
NODER: set of integer !nodene i nettverket
DAGSLOPSNODER: set of integer !dagsløpsnodene (D-node)
VLHNODER: set of integer !vedlikeholdsnoder (V-node)
STASJONER:  set of string !stasjoner

ADINNODER:  array(NODER,TYPE) of integer    !ant innoder per Dnode
AVINNODER:  array(NODER,TYPE) of integer    !ant innoder per Vnode

ADUTNODER:  array(NODER,TYPE) of integer    !ant utnoder per Dnode
AVUTNODER:  array(NODER,TYPE) of integer !ant innoder per Vnode

ANTALLNODER: integer !antall noder i nettverket
ANTALLDAGSLOP:  integer  !antall dagsløp i nettverket
ANTALLVLHNODER: integer  !antall vedlikehold i nettverket

M1: array(NODER,NODER, TYPE) of real !BIGM i restr: tellerdl
M3: array(NODER,NODER) of real  !BIGM i restr: tellervlh
M2: array(NODER,NODER) of real  !BIGM i restr: nullstill

KMGRENSE: array(TYPE) of integer  !km-grense for hver togtype


KMTT:  dynamic array(NODER,NODER) of real
!antall kilometer mellom hvert dagsløp


DAGSLOPSDAG: array(NODER) of integer !Hvilken dag hver node tilhører



DINNODER:  dynamic array(NODER,NODER,TYPE) of integer
!lovlige innoder for hvert dagsløp

VINNODER:  dynamic array(NODER,NODER,TYPE) of integer
!lovlige innoder for hvert vedlikehold

DUTNODER:  dynamic array(NODER,NODER,TYPE) of integer
!lovlige utnoder for hvert dagsløp

VUTNODER:  dynamic array(NODER,NODER,TYPE) of integer
!lovlige utnoder for hvert vedlikehold

LENGDE: array(DAGSLOPSNODER) of real  !lengde på dagsløp

MATKAP: array(TYPE) of integer  !materielltilgjengelighet

NODETYPE: array(NODER,TYPE) of integer
!en matrise som sier hvilke typer til hvilke noder,
!består av nullere og enere
!1=noden tilhører typen, 0 ellers

!AVHENGIG:  array(DAGSLOPSNODER, DAGSLOPSNODER) of integer
!sier hvilke dagsløp som må kjøres av samme type på grunn av avhengigheter
!hvis det står 1, så må de kjøres av samme togtype, men forskjellige subtyper.
```

```
!må lages på forhånd ved å se på turene i dagsløpene

NEDREGRENSE: real
TOMTOGLP: real

LOVLIG: integer
Subgrense: mpvar
Vedlikehold: mpvar
Tomtog: mpvar
x:  array(NODER,NODER,TYPE) of mpvar !kobling mellom to noder, flyt

Totalt: mpvar

!Restriksjoner
TOTALFLYTDAGSLOP: array(DAGSLOPSNODER,TYPE) of linctr
TOTALFLYTVHL: dynamic array(NODER,TYPE) of linctr
TOTALINNFLYTDAGSLOP: array(DAGSLOPSNODER) of linctr
TINNFLYTVEDLIKEHOLD: dynamic array(NODER) of linctr
Tilgjengelig: array(TYPE) of linctr


end-declarations

initializations from 'lovlig.dat'
LOVLIG
end-initializations


declarations

dualE1: array(NODER,NODER,TYPE,1..LOVLIG) of real
dualE2: array(NODER,NODER,TYPE,1..LOVLIG) of real
dualE3: array(NODER,NODER,TYPE,1..LOVLIG) of real
dualE4: array(NODER,TYPE,1..LOVLIG) of real

end-declarations

initializations from LOSNINGSUBLOVLIG
dualE1
dualE2
dualE3
dualE4
end-initializations

initializations from NETTVERK
AVINNODER
ADINNODER
AVUTNODER
ADUTNODER
VUTNODER
VINNODER
DUTNODER
DINNODER
TYPE
NODER
```

```
DAGSLOPSNODER
VLHNODER
ANTALLNODER
ANTALLDAGSLOP
ANTALLVLHNODER
KMTT
KMGRENSE
LENGDE
M1
M2
M3
UKE
TOMTOG
LIKEUKER
AVHENGIG
NODETYPE
MATKAP
DAGSLOPSDAG
end-initializations


initializations from "nedregrense.dat"
NEDREGRENSE
TOMTOGLP
end-initializations

declarations
TELLERe4: array(1..LOVLIG,TYPE,1..UKE) of real
Lovlighetskutt: array(1..LOVLIG, TYPE, 1..UKE) of linctr
AKTIV: array(TYPE,1..UKE,1..LOVLIG) of integer
end-declarations


initializations from "aktiv.dat"
AKTIV
end-initializations

LASTETID1 := gettime - STARTTIME

!-----------------------------------------------------
!lager beslutningsvarible og variabelrestiksjonene
!-----------------------------------------------------

forall(j in NODER|j<=ANTALLDAGSLOP, k in TYPE, u in 1..UKE) do
forall (a in 1..ADINNODER(j,k)) do

v:= j + ANTALLNODER*(u-1)
create(x(DINNODER(v,a,k),v,k))

end-do
end-do


forall(j in NODER|j>ANTALLDAGSLOP and j<=ANTALLNODER, k in TYPE, u in 1..UKE) do
forall(a in 1..AVINNODER(j,k)) do
```

```
v:= j+ ANTALLNODER*(u-1)
create(x(VINNODER(v,a,k),v,k))
end-do
end-do


forall(j in NODER|j<=ANTALLDAGSLOP, k in TYPE, u in 1..UKE) do
forall(a in 1..ADINNODER(j,k))do
v:= j + ANTALLNODER*(u-1)
x(DINNODER(v,a,k),v,k) is_binary
end-do
end-do

forall(j in NODER|j>ANTALLDAGSLOP and
j<=ANTALLNODER, k in TYPE,a in 1..AVINNODER(j,k), u in 1..UKE) do

v:= j+ ANTALLNODER*(u-1)
x(VINNODER(v,a,k),v,k)is_binary
end-do


!-----------------------------
! Flytrestriksjoner
!-----------------------------

forall(j in NODER|j<=ANTALLDAGSLOP, k in TYPE,u in 1..UKE)do

v:=j+ANTALLNODER*(u-1)
TOTALFLYTDAGSLOP(v,k):=
SUM(a in 1..ADINNODER(v,k)) x(DINNODER(v,a,k),v,k) -
SUM(b in 1..ADUTNODER(v,k)) x(v,DUTNODER(v,b,k),k)=0
end-do


forall(j in NODER|j>ANTALLDAGSLOP and j<=ANTALLNODER, k in TYPE, u in 1..UKE)do

v:=j+ANTALLNODER*(u-1)
TOTALFLYTVHL(v,k):=
SUM(a in 1..AVINNODER(v,k)) x(VINNODER(v,a,k),v,k) -
SUM(b in 1..AVUTNODER(v,k)) x(v,VUTNODER(v,b,k),k) =0

end-do

forall(j in NODER|j<=ANTALLDAGSLOP, u in 1..UKE)  do

NYj:=j+ANTALLNODER*(u-1)

TOTALINNFLYTDAGSLOP(NYj):=
 SUM(k in TYPE, a in 1..ADINNODER(NYj,k)) x(DINNODER(NYj,a,k),NYj,k)=1

end-do

forall(j in NODER|j>ANTALLDAGSLOP and j<=ANTALLNODER, u in 1..UKE) do
```

```
NYj:=j+ANTALLNODER *(u-1)

TINNFLYTVEDLIKEHOLD(NYj):=
SUM(k in TYPE, a in 1..AVINNODER(NYj,k))
x(VINNODER(NYj,a,k),NYj,k)<=1

end-do



!-------------------------------------------------------------------
!materiellltilgjengelighet, summerer antallet som krysser ei natt
!-------------------------------------------------------------------

forall(k in TYPE) do


Tilgjengelig(k) :=
sum(j in NODER|j<=ANTALLDAGSLOP, a in 1..ADINNODER(j,k)|
DAGSLOPSDAG(j)=2 and DAGSLOPSDAG(DINNODER(j,a,k))=1)
x(DINNODER(j,a,k),j,k)+

sum(j in NODER|j>ANTALLDAGSLOP and j<=ANTALLNODER,
a in 1..AVINNODER(j,k)|DAGSLOPSDAG(j)=2 and
DAGSLOPSDAG(VINNODER(j,a,k))=1) x(VINNODER(j,a,k),j,k) <=

MATKAP(k)

end-do



!-------------------------------------------------------------------
!Like uker hvis PTOTAL > TTOTAL og LIKEUKER=1
!Koblingene mellom søndag og mandag trenger ikke å være like uke for uke
!-------------------------------------------------------------------


if(LIKEUKER = 1) then
forall(u in 1..UKE|u>1) do

forall(k in TYPE, j in DAGSLOPSNODER|j<=ANTALLDAGSLOP, a in 1..ADINNODER(j,k),
i in DAGSLOPSNODER|DINNODER(j,a,k)= i)
if (not (DAGSLOPSDAG(i) = TTOTAL and DAGSLOPSDAG(j)= 1)) then

x(DINNODER(j+ANTALLNODER*(u-2),a,k),j+ANTALLNODER*(u-2),k) +
0.5* sum(b in 1..ADINNODER(j+ANTALLNODER*(u-2),k)|a<>b)
(x(DINNODER(j+ANTALLNODER*(u-2),a,k),DINNODER(j+ANTALLNODER*(u-2),b,k),k)+
x(DINNODER(j+ANTALLNODER*(u-2),b,k),j+ANTALLNODER*(u-2),k))-

(
x(DINNODER(j+ANTALLNODER*(u-1),a,k),j+ANTALLNODER*(u-1),k) +
0.5* sum(b in 1..ADINNODER(j+ANTALLNODER*(u-1),k)|a<>b)
(x(DINNODER(j+ANTALLNODER*(u-1),a,k),DINNODER(j+ANTALLNODER*(u-1),b,k),k)+
x(DINNODER(j+ANTALLNODER*(u-1),b,k),j+ANTALLNODER*(u-1),k)) ) = 0
```

```
end-if
end-do
end-if



(! ------------------------------------------
Avhengigheter mellom dagsløp, hvis det er mulig
å velge togtyper på dagsløpene
-----------------------------------------------!)
(!
forall(i in NODER|i <=ANTALLDAGSLOP, j in NODER|j<=ANTALLDAGSLOP)do

forall(u in 1..UKE) do
NYj := j + ANTALLNODER*(u-1)
NYi := i + ANTALLNODER*(u-1)

if(AVHENGIG(NYi,NYj)=1) then

!togtype 72 er egen hovedtype
sum(k in TYPE|k=72, a in 1..ADUTNODER(NYi,k)) x(NYi,DUTNODER(NYi,a,k),k) -
sum(k in TYPE|k=72, b in 1..ADUTNODER(NYj,k)) x(NYj,DUTNODER(NYj,b,k),k) = 0

sum(k in TYPE|k=692 or k=693, a in 1..ADUTNODER(NYi,k)) x(NYi,DUTNODER(NYi,a,k),k) -
sum(k in TYPE|k=692 or k=693, b in 1..ADUTNODER(NYj,k)) x(NYj,DUTNODER(NYj,b,k),k) = 0

end-if
end-do
end-do

!)
!------------------------------
!lovlighetskutt - restriksjon
!------------------------------

forall(a1 in 1..LOVLIG, k in TYPE, u in 1..UKE|AKTIV(k,u,a1)=0)do!|AKTIV(k,u,a1)=1) do

!alfa

forall( j in NODER|j <=ANTALLDAGSLOP, a in 1..ADINNODER(j,k), i in DAGSLOPSNODER|
i= DINNODER(j,a,k))do
TELLERe10(DINNODER(j+ANTALLNODER*(u-1),a,k),j+ANTALLNODER*(u-1),k,a1) :=

(M1(DINNODER(j+ANTALLNODER*(u-1),a,k),j+ANTALLNODER*(u-1),k) -

KMTT(DINNODER(j+ANTALLNODER*(u-1),a,k),j+ANTALLNODER*(u-1)) - LENGDE(j))*
dualE1(DINNODER(j+ANTALLNODER*(u-1),a,k),j+ANTALLNODER*(u-1),k,a1)


TELLERe1(DINNODER(j+ANTALLNODER*(u-1),a,k),j+ANTALLNODER*(u-1),k,a1) :=
floor (M1(DINNODER(j+ANTALLNODER*(u-1),a,k),j+ANTALLNODER*(u-1),k)*
dualE1(DINNODER(j+ANTALLNODER*(u-1),a,k),j+ANTALLNODER*(u-1),k,a1))*
x(DINNODER(j+ANTALLNODER*(u-1),a,k),j+ANTALLNODER*(u-1),k)
```

```
end-do

!beta
forall(j in NODER|j>ANTALLDAGSLOP and j<=ANTALLNODER, a in 1..AVINNODER(j,k)) do

TELLERe20(VINNODER(j+ANTALLNODER*(u-1),a,k),j+ANTALLNODER*(u-1),k,a1) :=

(M2(VINNODER(j+ANTALLNODER*(u-1),a,k),j+ANTALLNODER*(u-1))-

KMTT(VINNODER(j+ANTALLNODER*(u-1),a,k),j+ANTALLNODER*(u-1)) + KMGRENSE(k))*
dualE2(VINNODER(j+ANTALLNODER*(u-1),a,k),j+ANTALLNODER*(u-1),k,a1)


TELLERe2(VINNODER(j+ANTALLNODER*(u-1),a,k),j+ANTALLNODER*(u-1),k,a1) :=
floor (M2(VINNODER(j+ANTALLNODER*(u-1),a,k),j+ANTALLNODER*(u-1))*
dualE2(VINNODER(j+ANTALLNODER*(u-1),a,k),j+ANTALLNODER*(u-1),k,a1))*
x(VINNODER(j+ANTALLNODER*(u-1),a,k),j+ANTALLNODER*(u-1),k)



end-do

!gamma
forall (j in NODER|j>ANTALLDAGSLOP and j <=ANTALLNODER, a in 1..AVUTNODER(j,k))do

TELLERe30(j+ANTALLNODER*(u-1), VUTNODER(j+ANTALLNODER*(u-1),a,k),k,a1) :=

(M3(j+ANTALLNODER*(u-1),VUTNODER(j+ANTALLNODER*(u-1),a,k)) -

KMTT(j+ANTALLNODER*(u-1),VUTNODER(j+ANTALLNODER*(u-1),a,k)) -

LENGDE(VUTNODER(j+ANTALLNODER*(u-1),a,k)))*

dualE3(j+ANTALLNODER*(u-1),VUTNODER(j+ANTALLNODER*(u-1),a,k),k,a1)


TELLERe3(j+ANTALLNODER*(u-1), VUTNODER(j+ANTALLNODER*(u-1),a,k),k,a1) :=
        floor (M3(j+ANTALLNODER*(u-1),VUTNODER(j+ANTALLNODER*(u-1),a,k))*
dualE3(j+ANTALLNODER*(u-1),VUTNODER(j+ANTALLNODER*(u-1),a,k),k,a1))*
x(j+ANTALLNODER*(u-1),VUTNODER(j+ANTALLNODER*(u-1),a,k),k)


end-do

!delta
TELLERe40(a1,k,u) := SUM(j in NODER|j <=ANTALLDAGSLOP)
KMGRENSE(k)*dualE4(j+ANTALLNODER*(u-1),k,a1)



Lovlighetskutt(a1,k,u) :=
sum(j in NODER|j <=ANTALLDAGSLOP, a in 1..ADINNODER(j,k),
i in DAGSLOPSNODER |i = DINNODER(j,a,k))
```

```
TELLERe1(DINNODER(j+ANTALLNODER*(u-1),a,k),j+ANTALLNODER*(u-1),k,a1) +

sum(j in NODER|j>ANTALLDAGSLOP and j<=ANTALLNODER, a in 1..AVINNODER(j,k))
TELLERe2(VINNODER(j+ANTALLNODER*(u-1),a,k),j+ANTALLNODER*(u-1),k,a1) +

sum(j in NODER|j>ANTALLDAGSLOP and j <=ANTALLNODER, a in 1..AVUTNODER(j,k))
TELLERe3(j+ANTALLNODER*(u-1), VUTNODER(j+ANTALLNODER*(u-1),a,k),k,a1) >=


floor (sum(j in NODER|j <=ANTALLDAGSLOP, a in 1..ADINNODER(j,k),
 i in DAGSLOPSNODER |i = DINNODER(j,a,k))

TELLERe10(DINNODER(j+ANTALLNODER*(u-1),a,k),j+ANTALLNODER*(u-1),k,a1) +

sum(j in NODER|j>ANTALLDAGSLOP and j<=ANTALLNODER, a in 1..AVINNODER(j,k))
TELLERe20(VINNODER(j+ANTALLNODER*(u-1),a,k),j+ANTALLNODER*(u-1),k,a1) +

sum(j in NODER|j>ANTALLDAGSLOP and j <=ANTALLNODER, a in 1..AVUTNODER(j,k))
TELLERe30(j+ANTALLNODER*(u-1), VUTNODER(j+ANTALLNODER*(u-1),a,k),k,a1)+

TELLERe40(a1,k,u))


end-do



!-----------------------------------------
!Målfunksjon
!-----------------------------------------




Vedlikehold = SUM(j in NODER|j>ANTALLDAGSLOP and
j<=ANTALLNODER, k in TYPE, a in 1..AVINNODER(j,k))
x(VINNODER(j,a,k),j,k)+

SUM(j in NODER|j>ANTALLDAGSLOP and
j<=ANTALLNODER, k in TYPE, a in 1..AVINNODER(j,k), u in 1..UKE|u>1)

x(VINNODER(j+ANTALLNODER*(u-1),a,k),j+ANTALLNODER*(u-1),k)

Vedlikehold >= sum(k in TYPE)ceil(sum(i in NODER|i <= ANTALLDAGSLOP)
(NODETYPE(i,k)* LENGDE(i) /KMGRENSE(k))*UKE)

Vedlikehold >= ceil(NEDREGRENSE)


Tomtog =
SUM(k in TYPE,j in NODER|j<= ANTALLDAGSLOP, a in 1..ADINNODER(j,k))
KMTT(DINNODER(j,a,k),j)*x(DINNODER(j,a,k),j,k)+

SUM(k in TYPE, j in NODER|j>ANTALLDAGSLOP and j<=ANTALLNODER,
```

```
a in 1..AVINNODER(j,k))
KMTT(VINNODER(j,a,k),j)*x(VINNODER(j,a,k),j,k)+

SUM(k in TYPE, j in NODER|j<= ANTALLDAGSLOP, a in 1..ADINNODER(j,k), u in 1..UKE|u>1)
KMTT(DINNODER(j+ANTALLNODER*(u-1),a,k),j+ANTALLNODER*(u-1))*
x(DINNODER(j+ANTALLNODER*(u-1),a,k),j+ANTALLNODER*(u-1),k)+

SUM(k in TYPE, j in NODER|j>ANTALLDAGSLOP and j<=ANTALLNODER,
a in 1..AVINNODER(j,k),u in 1..UKE|u>1)
KMTT(VINNODER(j+ANTALLNODER*(u-1),a,k),j+ANTALLNODER*(u-1))*
x(VINNODER(j+ANTALLNODER*(u-1),a,k),j+ANTALLNODER*(u-1),k)

!Tomtog>= TOMTOGLP



Totalt = Vedlikehold + 1/100000*Tomtog




!-------------------------------------------------
! Setter en nedre grense på målfunksjonen
!-------------------------------------------------



minimize(Totalt)

LOSNINGSTID1 := gettime - STARTTIME
STARTSKRIV := gettime


!--------------------------
!Utskrift
!--------------------------

!skrive til fil

fopen("losning_master.dat",F_OUTPUT)

        writeln("LB:[", getobjval, "]")

        writeln('x:[')
        forall(k in TYPE, j in NODER|j <= ANTALLDAGSLOP, a in 1..ADINNODER(j,k),
u in 1..UKE)do

        writeln(" (",DINNODER(j+ANTALLNODER*(u-1),a,k)," ",j+ANTALLNODER*(u-1)," ",k,") " ,
        getsol(x(DINNODER(j+ANTALLNODER*(u-1),a,k),j+ANTALLNODER*(u-1),k)))
        end-do


        forall(k in TYPE, j in NODER|j > ANTALLDAGSLOP and j<=ANTALLNODER,
```

```
a in 1..AVINNODER(j,k), u in 1..UKE)do

        writeln(" (",VINNODER(j+ANTALLNODER*(u-1),a,k)," ",j+ANTALLNODER*(u-1)," ",k,") " ,
        getsol(x(VINNODER(j+ANTALLNODER*(u-1),a,k),j+ANTALLNODER*(u-1),k)))
        end-do


        writeln(']')
fclose(F_OUTPUT)

 writeln("LB:[", getobjval, "]")


fopen("losning_master_objekt.dat",F_APPEND)
writeln("LB:[", getobjval, "]")
writeln('x:[')
        forall(k in TYPE, j in NODER|j <= ANTALLDAGSLOP, a in 1..ADINNODER(j,k),
u in 1..UKE)do

        writeln(" (",DINNODER(j+ANTALLNODER*(u-1),a,k)," ",j+ANTALLNODER*(u-1)," ",k,") " ,
        getsol(x(DINNODER(j+ANTALLNODER*(u-1),a,k),j+ANTALLNODER*(u-1),k)))
        end-do


        forall(k in TYPE, j in NODER|j > ANTALLDAGSLOP and j<=ANTALLNODER,
a in 1..AVINNODER(j,k), u in 1..UKE)do

        writeln(" (",VINNODER(j+ANTALLNODER*(u-1),a,k)," ",j+ANTALLNODER*(u-1)," ",k,") " ,
        getsol(x(VINNODER(j+ANTALLNODER*(u-1),a,k),j+ANTALLNODER*(u-1),k)))
        end-do


        writeln(']')
fclose(F_APPEND)



procedure print_status

declarations
status: array({XPRS_OPT,XPRS_UNF,XPRS_INF,XPRS_UNB,XPRS_OTH}) of string
end-declarations

status := ["Optimum", "Unfinished", "Infeasible",
"Unbounded", "Failed"]
!writeln("Problem status: ", status(getprobstat))
fopen('status_master.dat', F_OUTPUT)
writeln("STATUS_MASTER: ", status(getprobstat))
fclose(F_OUTPUT)
end-procedure

print_status
```

```
(!
Kode for å kommunisere med batchfilen....
!)

declarations
STATUS_MASTER: string
returtall: integer

LB: real
end-declarations

initializations from 'status_master.dat'
STATUS_MASTER
end-initializations


!setter verdier til variablene
LB := 0
returtall := 0


if (STATUS_MASTER = "Optimum") then
LB := getobjval
fopen('objekt_master.dat', F_OUTPUT)
writeln('LB : ', LB)
fclose(F_OUTPUT)
writeln('Lovlig: ', LOVLIG)
returtall :=5
if(LOVLIG = 0) then
returtall := 8
end-if
elif (STATUS_MASTER = "Infeasible") then
writeln("Uløselig masterproblem - Uløselig problem ")
returtall := 9
else
returtall := 9

end-if

writeln(returtall)
writeln('Vedlikehold: ',getsol(Vedlikehold))


declarations
LOSNINGSTID: real
SKRIVETID: real
LASTETID: real
end-declarations

initializations from "losningstid_master.dat"
LOSNINGSTID
SKRIVETID
LASTETID
end-initializations
```

```
SKRIVETID1 := gettime - STARTSKRIV

fopen("losningstid_master.dat", F_OUTPUT)
writeln("LOSNINGSTID :", LOSNINGSTID+LOSNINGSTID1)
writeln("SKRIVETID :", SKRIVETID+SKRIVETID1)
writeln("LASTETID :", LASTETID + LASTETID1)
fclose(F_OUTPUT)



exit(returtall)

end-model
```

C.2 Fase1-problemet

```
model NSB_bender
uses "mmxprs"
uses "mmsystem"

(!-----------------------------------------------------
Fase1-modell til Benders algoritme, for  vedlikeholdsproblemet

Laget av Marianne Risberg
Masteroppgave NTNU våren 2005
-----------------------------------------------------!)



parameters

LOSNINGMASTER = 'losning_master.dat'
NETTVERK  = 'generert_graf.dat'
end-parameters

STARTTIME := gettime

declarations
UKE: integer
TOMTOG: integer
LIKEUKER: integer
TYPE: set of integer !togtyper
NODER: set of integer !nodene i nettverket
DAGSLOPSNODER: set of integer !dagsløpsnodene (D-node)
VLHNODER: set of integer !vedlikeholdsnoder (V-node)
STASJONER:  set of string !stasjoner

ADINNODER:  array(NODER,TYPE) of integer    !ant innoder per Dnode
AVINNODER:  array(NODER,TYPE) of integer    !ant innoder per Vnode
```

```
ADUTNODER:  array(NODER,TYPE) of integer    !ant utnoder per Dnode
AVUTNODER:  array(NODER,TYPE) of integer !ant innoder per Vnode

ANTALLNODER: integer !antall noder i nettverket
ANTALLDAGSLOP:  integer  !antall dagsløp i nettverket
ANTALLVLHNODER: integer  !antall vedlikehold i nettverket

M1: array(NODER,NODER, TYPE) of real !BIGM i restr: tellerdl
M2: array(NODER,NODER) of real  !BIGM i restr: tellervlh
M3: array(NODER,NODER) of real  !BIGM i restr: nullstill

KMGRENSE: array(TYPE) of integer  !km-grense for hver togtype

KMTT:  dynamic array(NODER,NODER) of real
!antall kilometer mellom hvert dagsløp

DINNODER:  dynamic array(NODER,NODER,TYPE) of integer
!lovlige innoder for hvert dagsløp

VINNODER:  dynamic array(NODER,NODER,TYPE) of integer
!lovlige innoder for hvert vedlikehold

DUTNODER:  dynamic array(NODER,NODER,TYPE) of integer
!lovlige utnoder for hvert dagsløp

VUTNODER:  dynamic array(NODER,NODER,TYPE) of integer
!lovlige utnoder for hvert vedlikehold

LENGDE: array(DAGSLOPSNODER) of real  !lengde på dagsløp


NODETYPE: array(NODER,TYPE) of integer
!en matrise som sier hvilke typer til hvilke noder,
!består av nullere og enere
!1=noden tilhører typen, 0 ellers


kuttmodell: integer

!Beslutningsvariable:

x:  array(NODER,NODER,TYPE) of real !kobling mellom to noder, flyt, fra master
z: array(NODER,TYPE) of mpvar !teller kilometer

v2: array(NODER,NODER,TYPE) of mpvar
v4: array(NODER,NODER,TYPE) of mpvar
v6: array(NODER,NODER,TYPE) of mpvar
v8: array(NODER,TYPE) of mpvar
Total: mpvar

LOVLIG: integer

end-declarations
```

```
initializations from 'lovlig.dat'
LOVLIG
end-initializations

initializations from NETTVERK
ADUTNODER
ADINNODER
AVUTNODER
AVINNODER
DINNODER
DUTNODER
VINNODER
VUTNODER
TYPE
UKE
TOMTOG
LIKEUKER
NODER
DAGSLOPSNODER
ANTALLNODER
ANTALLDAGSLOP
ANTALLVLHNODER
LENGDE
KMGRENSE
KMTT
M1
M2
M3
end-initializations


declarations

dualE1: array(NODER,NODER,TYPE, 1..LOVLIG) of real
dualE2: array(NODER,NODER,TYPE, 1..LOVLIG) of real
dualE3: array(NODER,NODER,TYPE, 1..LOVLIG) of real
dualE4: array(NODER,TYPE, 1..LOVLIG) of real

end-declarations



initializations from LOSNINGMASTER
x
end-initializations




!-----------------------------------------------------
!lager beslutningsvarible og variabelrestiksjonene
!-----------------------------------------------------
```

```
forall(i in NODER|i <=ANTALLDAGSLOP,k in TYPE) do
create(z(i,k))

forall(u in 1..UKE|u>1) do
v:= i + ANTALLNODER*(u-1)
   create(z(v,k))
end-do

end-do

forall(u in 1..UKE, j in NODER|j<=ANTALLDAGSLOP,k in TYPE, a in 1..ADINNODER(j,k),
i in DAGSLOPSNODER|i=DINNODER(j,a,k))do

create(v2(DINNODER(j+ANTALLNODER*(u-1),a,k),j+ANTALLNODER*(u-1),k))
end-do



forall(u in 1..UKE, j in NODER|j>ANTALLDAGSLOP and j<=ANTALLNODER,k in TYPE,
a in 1..AVUTNODER(j,k))do

create(v4(j+ANTALLNODER*(u-1),VUTNODER(j+ANTALLNODER*(u-1),a,k),k))
end-do


forall(u in 1..UKE, j in NODER|j>ANTALLDAGSLOP and j<=ANTALLNODER, k in TYPE,
a in 1..AVINNODER(j,k)) do
create(v6(VINNODER(j+ANTALLNODER*(u-1),a,k),j+ANTALLNODER*(u-1),k))
end-do



forall(u in 1..UKE, i in NODER|i<=ANTALLDAGSLOP, k in TYPE) do
create(v8(i+ANTALLNODER*(u-1),k))
end-do



forall(i in NODER|i<=ANTALLDAGSLOP, k in TYPE) do

forall(u in 1..UKE) do
v:= i + ANTALLNODER*(u-1)
Grensez(v,k) := z(v,k)-v8(v,k)<= KMGRENSE(k)
end-do

end-do




!-----------------------------------
!Teller når det er dagslop til dagslop
!-----------------------------------
```

```
forall(u in 1..UKE, j in NODER|j<=ANTALLDAGSLOP,k in TYPE, a in 1..ADINNODER(j,k),
i in DAGSLOPSNODER|i = DINNODER(j,a,k))do

Tellerkm(DINNODER(j+ANTALLNODER*(u-1),a,k),j+ANTALLNODER*(u-1),k):=
z(DINNODER(j+ANTALLNODER*(u-1),a,k),k)-z(j+ANTALLNODER*(u-1),k)-
v2(DINNODER(j+ANTALLNODER*(u-1),a,k),j+ANTALLNODER*(u-1),k)<= -
M1(DINNODER(j+ANTALLNODER*(u-1),a,k),j+ANTALLNODER*(u-1),k) *
x(DINNODER(j+ANTALLNODER*(u-1),a,k),j+ANTALLNODER*(u-1),k)+
M1(DINNODER(j+ANTALLNODER*(u-1),a,k),j+ANTALLNODER*(u-1),k) -
KMTT(DINNODER(j+ANTALLNODER*(u-1),a,k),j+ANTALLNODER*(u-1))-LENGDE(j)


  end-do

!-----------------------------------
! Fra vedlikehold til dagsløp
!-----------------------------------
forall(u in 1..UKE, j in NODER|j>ANTALLDAGSLOP and
j<=ANTALLNODER,k in TYPE,a in 1..AVUTNODER(j,k))do

VDTellerkm(j+ANTALLNODER*(u-1),VUTNODER(j+ANTALLNODER*(u-1),a,k),k):=
-z(VUTNODER(j+ANTALLNODER*(u-1),a,k),k) -
v4(j+ANTALLNODER*(u-1),VUTNODER(j+ANTALLNODER*(u-1),a,k),k)<= -
M3(j+ANTALLNODER*(u-1),VUTNODER(j+ANTALLNODER*(u-1),a,k)) *
x(j+ANTALLNODER*(u-1), VUTNODER(j+ANTALLNODER*(u-1),a,k),k)+
M3(j+ANTALLNODER*(u-1),VUTNODER(j+ANTALLNODER*(u-1),a,k)) -
KMTT(j+ANTALLNODER*(u-1),VUTNODER(j+ANTALLNODER*(u-1),a,k)) -
LENGDE(VUTNODER(j,a,k))


end-do

!---------------------------------------
! fra dagsløp til vedlikehold
!---------------------------------------
forall(u in 1..UKE, j in NODER|j>ANTALLDAGSLOP and
j<=ANTALLNODER, k in TYPE, a in 1..AVINNODER(j,k)) do

TellerkmVlh(VINNODER(j+ANTALLNODER*(u-1),a,k),j+ANTALLNODER*(u-1),k):=
z(VINNODER(j+ANTALLNODER*(u-1),a,k),k)-
v6(VINNODER(j+ANTALLNODER*(u-1),a,k),j+ANTALLNODER*(u-1),k) <= -
M2(VINNODER(j+ANTALLNODER*(u-1),a,k),j+ANTALLNODER*(u-1))*
x(VINNODER(j+ANTALLNODER*(u-1),a,k),j+ANTALLNODER*(u-1),k)+
M2(VINNODER(j+ANTALLNODER*(u-1),a,k),j+ANTALLNODER*(u-1))-
KMTT(VINNODER(j+ANTALLNODER*(u-1),a,k),j+ANTALLNODER*(u-1))+ KMGRENSE(k)


end-do



!-----------------------------------------
!Målfunksjon
!-----------------------------------------
```

```
Total =
SUM(u in 1..UKE, j in NODER|j<=ANTALLDAGSLOP,k in TYPE, a in 1..ADINNODER(j,k),
i in DAGSLOPSNODER|i=DINNODER(j,a,k))

v2(DINNODER(j+ANTALLNODER*(u-1),a,k),j+ANTALLNODER*(u-1),k) +

SUM(u in 1..UKE, j in NODER|j>ANTALLDAGSLOP and j<=ANTALLNODER,
k in TYPE,a in 1..AVUTNODER(j,k))

 v4(j+ANTALLNODER*(u-1),VUTNODER(j+ANTALLNODER*(u-1),a,k),k)+

SUM(u in 1..UKE, j in NODER|j>ANTALLDAGSLOP and j<=ANTALLNODER,
k in TYPE, a in 1..AVINNODER(j,k))

 v6(VINNODER(j+ANTALLNODER*(u-1),a,k),j+ANTALLNODER*(u-1),k)+

SUM(u in 1..UKE, j in NODER|j<=ANTALLDAGSLOP, k in TYPE)
v8(j+ANTALLNODER*(u-1),k)


minimize(Total)


LOSNINGSTID1 := gettime - STARTTIME
STARTSKRIV := gettime


if(getobjval > 0)then

writeln("må kutte, løs master igjen")


if(LOVLIG>0) then
initializations from 'losning_subproblemstraale.dat'
dualE1
dualE2
dualE3
dualE4
end-initializations


end-if

LOVLIG := LOVLIG +1

fopen('lovlig.dat',F_OUTPUT)
writeln("LOVLIG: ", LOVLIG)
fclose(F_OUTPUT)


forall(l in 1..LOVLIG|l=LOVLIG, j in NODER|j<=ANTALLDAGSLOP,k in TYPE,
u in 1..UKE, a in 1..ADINNODER(j,k), i in DAGSLOPSNODER|i=DINNODER(j,a,k))do
```

```
DdualE1(DINNODER(j+ANTALLNODER*(u-1),a,k),j+ANTALLNODER*(u-1),k,l) :=
getdual(Tellerkm(DINNODER(j+ANTALLNODER*(u-1),a,k),j+ANTALLNODER*(u-1),k))
end-do

forall(l in 1..LOVLIG|l=LOVLIG, j in NODER|j>ANTALLDAGSLOP and
j<=ANTALLNODER, k in TYPE, a in 1..AVINNODER(j,k), u in 1..UKE) do

DdualE2(VINNODER(j+ANTALLNODER*(u-1),a,k),j+ANTALLNODER*(u-1),k,l) :=
getdual(TellerkmVlh(VINNODER(j+ANTALLNODER*(u-1),a,k),j+ANTALLNODER*(u-1),k))
end-do

forall(l in 1..LOVLIG|l=LOVLIG, j in NODER|j>ANTALLDAGSLOP and
j<=ANTALLNODER, k in TYPE,a in 1..AVUTNODER(j,k), u in 1..UKE)do

DdualE3(j,VUTNODER(j+ANTALLNODER*(u-1),a,k),k,l) :=
getdual(VDTellerkm(j+ANTALLNODER*(u-1),VUTNODER(j+ANTALLNODER*(u-1),a,k),k))
end-do

forall(l in 1..LOVLIG|l=LOVLIG, i in NODER|i<=ANTALLDAGSLOP, k in TYPE, u in 1..UKE) do
DdualE4(i+ANTALLNODER*(u-1),k,l) := getdual(Grensez(i+ANTALLNODER*(u-1),k))
end-do


fopen("losning_subproblemstraale.dat",F_OUTPUT)

writeln("dualE1 :[")
forall(k in TYPE,l in 1..LOVLIG, u in 1..UKE, j in NODER|j <= ANTALLDAGSLOP,
a in 1..ADINNODER(j,k), i in DAGSLOPSNODER|i =DINNODER(j,a,k))do

        if(l < LOVLIG) then
        writeln(" (",DINNODER(j+ANTALLNODER*(u-1),a,k)," ",j+ANTALLNODER*(u-1)," ",k," ", l,") "
,dualE1(DINNODER(j+ANTALLNODER*(u-1),a,k),j+ANTALLNODER*(u-1),k,l))

elif(l=LOVLIG) then
        writeln(" (",DINNODER(j+ANTALLNODER*(u-1),a,k)," ",j+ANTALLNODER*(u-1)," ",k," ", l,") "
,DdualE1(DINNODER(j+ANTALLNODER*(u-1),a,k),j+ANTALLNODER*(u-1),k,l))
  end-if
  end-do
    writeln("]")

    writeln("dualE2 :[")
  forall(k in TYPE, l in 1..LOVLIG, j in NODER|j>ANTALLDAGSLOP and j<=ANTALLNODER,
a in 1..AVINNODER(j,k), u in 1..UKE)do

if(l < LOVLIG) then
writeln(" (",VINNODER(j+ANTALLNODER*(u-1),a,k)," ",j+ANTALLNODER*(u-1)," ",k," ", l,") "
,dualE2(VINNODER(j+ANTALLNODER*(u-1),a,k),j+ANTALLNODER*(u-1),k,l))

elif(l = LOVLIG) then
   writeln(" (",VINNODER(j+ANTALLNODER*(u-1),a,k)," ",j+ANTALLNODER*(u-1)," ",k," ", l,") "
,DdualE2(VINNODER(j+ANTALLNODER*(u-1),a,k),j+ANTALLNODER*(u-1),k,l))
end-if
  end-do
    writeln("]")
```

```
    writeln("dualE3 :[")
    forall(l in 1..LOVLIG, j in NODER|j>ANTALLDAGSLOP and j<=ANTALLNODER,k in TYPE,
    a in 1..AVUTNODER(j,k), u in 1..UKE)do
if(l<LOVLIG) then
writeln( " (",j+ANTALLNODER*(u-1)," ",VUTNODER(j+ANTALLNODER*(u-1),a,k)," ",k," ",l, ") " ,
dualE3(j,VUTNODER(j+ANTALLNODER*(u-1),a,k),k,l))
elif(l=LOVLIG) then
writeln( " (",j+ANTALLNODER*(u-1)," ",VUTNODER(j+ANTALLNODER*(u-1),a,k)," ",k," ",l, ") " ,
DdualE3(j,VUTNODER(j+ANTALLNODER*(u-1),a,k),k,l))
end-if
end-do
writeln("]")

    writeln("dualE4 :[")
    forall(k in TYPE, l in 1..LOVLIG, j in NODER|j<=ANTALLDAGSLOP, u in 1..UKE)do
        if(l<LOVLIG) then
        writeln(" (",j+ANTALLNODER*(u-1)," ",k," ", l,") " ,dualE4(j+ANTALLNODER*(u-1),k,l))
        elif(l=LOVLIG) then
        writeln(" (",j+ANTALLNODER*(u-1)," ",k," ", l,") " ,DdualE4(j+ANTALLNODER*(u-1),k,l))
        end-if
    end-do
    writeln("]")

fclose(F_OUTPUT)




returtallet := 4


fopen("kuttall.dat", F_OUTPUT)

fclose(F_OUTPUT)

fopen("losning_fase1.dat", F_APPEND)
writeln("objekt: ", getobjval)
fclose(F_APPEND)


elif(getobjval=0) then
writeln("løs subproblem")
returtallet := 6

end-if

writeln(getobjval)

declarations
LOSNINGSTID: real
SKRIVETID: real
end-declarations
```

```
initializations from "losningstid_fase1.dat"
LOSNINGSTID
SKRIVETID
end-initializations

SKRIVETID1 := gettime - STARTSKRIV

fopen("losningstid_fase1.dat", F_OUTPUT)
writeln("LOSNINGSTID :", LOSNINGSTID+LOSNINGSTID1)
writeln("SKRIVETID :", SKRIVETID+SKRIVETID1)
fclose(F_OUTPUT)




exit(returtallet)

end-model
```

C.3 Sub-problem

```
model NSB_bender
uses "mmxprs"
uses "mmsystem"

(!-----------------------------------------------------
Subproblemet til benders algoritme for vedlikeholdsproblemet.

Laget av Marianne Risberg
Masteroppgave NTNU våren 2005
-----------------------------------------------------!)



parameters
LOSNINGSUB = 'losning_subproblem.dat'
LOSNINGMASTER = 'losning_master.dat'
NETTVERK  = 'generert_graf.dat'
end-parameters

STARTTIME := gettime


declarations


UKE: integer
TOMTOG: integer
LIKEUKER: integer
DAG:  set of integer !dager
TYPE: set of integer !togtyper
NODER: set of integer !nodene i nettverket
```

```
DAGSLOPSNODER: set of integer !dagsløpsnodene (D-node)
VLHNODER: set of integer !vedlikeholdsnoder (V-node)
STASJONER:  set of string !stasjoner

ADINNODER:  array(NODER,TYPE) of integer    !ant innoder per Dnode
AVINNODER:  array(NODER,TYPE) of integer    !ant innoder per Vnode

ADUTNODER:  array(NODER,TYPE) of integer    !ant utnoder per Dnode
AVUTNODER:  array(NODER,TYPE) of integer !ant innoder per Vnode

ANTALLNODER: integer !antall noder i nettverket
ANTALLDAGSLOP:  integer  !antall dagsløp i nettverket
ANTALLVLHNODER: integer  !antall vedlikehold i nettverket

M1: array(NODER,NODER, TYPE) of real !BIGM i restr: tellerdl
M2: array(NODER,NODER) of real  !BIGM i restr: tellervlh
M3: array(NODER,NODER) of real  !BIGM i restr: nullstill

KMGRENSE: array(TYPE) of integer  !km-grense for hver togtype


KM: array(STASJONER,STASJONER) of real
!antall kilometer mellom hver stasjon


KMTT:  dynamic array(NODER,NODER) of real
!antall kilometer mellom hvert dagsløp

STARTTID:  array(NODER) of real  !starttid for hver node

SLUTTID:  array(NODER) of real  !sluttid for hver node

STARTSTED:  array(NODER) of string  !startsted for hver node

SLUTTSTED:  array(NODER) of string  !sluttsted for hver node

DAGSLOPSDAG: array(NODER) of integer !Hvilken dag hver node tilhører


TID : array(STASJONER,STASJONER) of real
!tiden det tar å kjøre mellom to stasjoner

TIDTT : dynamic array(NODER,NODER) of real
!tiden det tar å kjøre mellom to noder

DINNODER:  dynamic array(NODER,NODER,TYPE) of integer
!lovlige innoder for hvert dagsløp

VINNODER:  dynamic array(NODER,NODER,TYPE) of integer
!lovlige innoder for hvert vedlikehold

DUTNODER:  dynamic array(NODER,NODER,TYPE) of integer
!lovlige utnoder for hvert dagsløp

VUTNODER:  dynamic array(NODER,NODER,TYPE) of integer
```

```
!lovlige utnoder for hvert vedlikehold

LENGDE: array(DAGSLOPSNODER) of real  !lengde på dagsløp


NODETYPE: array(NODER,TYPE) of integer
!en matrise som sier hvilke typer til hvilke noder,
!består av nullere og enere
!1=noden tilhører typen, 0 ellers


!AVHENGIG:  array(DAGSLOPSNODER, DAGSLOPSNODER) of integer
!sier hvilke dagsløp som må kjøres av samme type på grunn av avhengigheter
!hvis det står 1, så må de kjøres av samme togtype, men forskjellige subtyper.
!må lages på forhånd ved å se på turene i dagsløpene


!Beslutningsvariable:

x:  array(NODER,NODER,TYPE) of integer !kobling mellom to noder, flyt
!denne bestemmes av masterproblemet.
z: array(NODER,TYPE) of mpvar !teller kilometer

LOVLIG: integer

end-declarations


initializations from 'lovlig.dat'
LOVLIG
end-initializations



initializations from NETTVERK
ADUTNODER
ADINNODER
AVUTNODER
AVINNODER

DINNODER
DUTNODER
VINNODER
VUTNODER

UKE
LIKEUKER
TOMTOG
NODER
TYPE
DAGSLOPSNODER
LENGDE
KMTT
KMGRENSE
NODETYPE
```

```
ANTALLDAGSLOP
ANTALLNODER
ANTALLVLHNODER
M1
M2
M3

end-initializations


initializations from LOSNINGMASTER
x
end-initializations









forall(i in NODER|i <=ANTALLDAGSLOP,k in TYPE) do
forall(u in 1..UKE) do
v:= i + ANTALLNODER*(u-1)
   create(z(v,k))
end-do
end-do


forall(i in NODER|i<=ANTALLDAGSLOP, k in TYPE) do
forall(u in 1..UKE) do
v:= i + ANTALLNODER*(u-1)
Grensez(v,k) := z(v,k)<= KMGRENSE(k)
end-do
end-do



!-----------------------------------
!Teller når det er dagslop til dagslop
!-----------------------------------


forall(u in 1..UKE, j in NODER|j<=ANTALLDAGSLOP,k in TYPE, a in 1..ADINNODER(j,k),
i in DAGSLOPSNODER|i=DINNODER(j,a,k))do


Tellerkm(DINNODER(j+ANTALLNODER*(u-1),a,k),j+ANTALLNODER*(u-1),k):=
z(DINNODER(j+ANTALLNODER*(u-1),a,k),k)-z(j+ANTALLNODER*(u-1),k)<= -
M1(DINNODER(j+ANTALLNODER*(u-1),a,k),j+ANTALLNODER*(u-1),k) *
x(DINNODER(j+ANTALLNODER*(u-1),a,k),j+ANTALLNODER*(u-1),k)+
M1(DINNODER(j+ANTALLNODER*(u-1),a,k),j+ANTALLNODER*(u-1),k) -
KMTT(DINNODER(j+ANTALLNODER*(u-1),a,k),j+ANTALLNODER*(u-1))-LENGDE(j)
```

```
  end-do

!----------------------------------
! Fra vedlikehold til dagsløp
!----------------------------------
forall(u in 1..UKE, j in NODER|j>ANTALLDAGSLOP and
j<=ANTALLNODER,k in TYPE,a in 1..AVUTNODER(j,k))do

VDTellerkm(j+ANTALLNODER*(u-1),VUTNODER(j+ANTALLNODER*(u-1),a,k),k):=
-z(VUTNODER(j+ANTALLNODER*(u-1),a,k),k)<= -
M3(j+ANTALLNODER*(u-1),VUTNODER(j+ANTALLNODER*(u-1),a,k)) *
x(j+ANTALLNODER*(u-1), VUTNODER(j+ANTALLNODER*(u-1),a,k),k)+
M3(j+ANTALLNODER*(u-1),VUTNODER(j+ANTALLNODER*(u-1),a,k)) -
KMTT(j+ANTALLNODER*(u-1),VUTNODER(j+ANTALLNODER*(u-1),a,k)) - LENGDE(VUTNODER(j,a,k))

end-do

!---------------------------------------
! fra dagsløp til vedlikehold
!---------------------------------------
forall(u in 1..UKE, j in NODER|j>ANTALLDAGSLOP and
j<=ANTALLNODER, k in TYPE, a in 1..AVINNODER(j,k)) do

TellerkmVlh(VINNODER(j+ANTALLNODER*(u-1),a,k),j+ANTALLNODER*(u-1),k):=
z(VINNODER(j+ANTALLNODER*(u-1),a,k),k) <= -
M2(VINNODER(j+ANTALLNODER*(u-1),a,k),j+ANTALLNODER*(u-1))*
x(VINNODER(j+ANTALLNODER*(u-1),a,k),j+ANTALLNODER*(u-1),k)+
M2(VINNODER(j+ANTALLNODER*(u-1),a,k),j+ANTALLNODER*(u-1))-
KMTT(VINNODER(j+ANTALLNODER*(u-1),a,k),j+ANTALLNODER*(u-1))+ KMGRENSE(k)


end-do




!-----------------------------------------
!Målfunksjon
!-----------------------------------------

Vedlikehold := SUM(j in NODER|j>ANTALLDAGSLOP and
j<=ANTALLNODER, k in TYPE, a in 1..AVINNODER(j,k))
x(VINNODER(j,a,k),j,k)+

SUM(j in NODER|j>ANTALLDAGSLOP and
j<=ANTALLNODER, k in TYPE, a in 1..AVINNODER(j,k), u in 1..UKE|u>1)

x(VINNODER(j+ANTALLNODER*(u-1),a,k),j+ANTALLNODER*(u-1),k)



Tomtog :=
SUM(k in TYPE,j in NODER|j<= ANTALLDAGSLOP, a in 1..ADINNODER(j,k))
KMTT(DINNODER(j,a,k),j)*x(DINNODER(j,a,k),j,k)+
```

```
SUM(k in TYPE, j in NODER|j>ANTALLDAGSLOP and j<=ANTALLNODER,
a in 1..AVINNODER(j,k))
KMTT(VINNODER(j,a,k),j)*x(VINNODER(j,a,k),j,k)+

SUM(k in TYPE, j in NODER|j<= ANTALLDAGSLOP, a in 1..ADINNODER(j,k), u in 1..UKE|u>1)
KMTT(DINNODER(j+ANTALLNODER*(u-1),a,k),j+ANTALLNODER*(u-1))*
x(DINNODER(j+ANTALLNODER*(u-1),a,k),j+ANTALLNODER*(u-1),k)+

SUM(k in TYPE, j in NODER|j>ANTALLDAGSLOP and j<=ANTALLNODER,
a in 1..AVINNODER(j,k),u in 1..UKE|u>1)
KMTT(VINNODER(j+ANTALLNODER*(u-1),a,k),j+ANTALLNODER*(u-1))*
x(VINNODER(j+ANTALLNODER*(u-1),a,k),j+ANTALLNODER*(u-1),k)



Subproblem := sum(u in 1..UKE ,j in NODER|j<= ANTALLDAGSLOP, k in TYPE)
 z(j+ANTALLNODER*(u-1),k)


Total := Subproblem



minimize(Total)
LOSNINGSTID:= gettime - STARTTIME

STARTSKRIV := gettime

fopen("losning_subproblem.dat", F_OUTPUT)

writeln("Problemet er løst til optimalitet")

writeln("Lovlig løsning funnet til subproblemet")
writeln("Vedlikehold: ", Vedlikehold)
writeln("Tomtog: ", Tomtog)

forall (j in NODER|j <=ANTALLDAGSLOP, u in 1..UKE, k in TYPE)do
writeln("z(",j+ANTALLNODER*(u-1), ",",k,")  =", getsol(z(j+ANTALLNODER*(u-1),k)))
end-do

fclose(F_OUTPUT)




writeln("Problemet er løst til optimalitet")

writeln("Lovlig løsning funnet til subproblemet")
writeln("Vedlikehold: ", Vedlikehold)
writeln("Tomtog: ", Tomtog)
```

```
SKRIVETID := gettime - STARTSKRIV

fopen("losningstid_subproblem.dat", F_OUTPUT)
writeln("LOSNINGSTID :", LOSNINGSTID)
writeln("SKRIVETID :", SKRIVETID)
fclose(F_OUTPUT)

returtall := 9
exit(returtall)
end-model
```

Vedlegg D: Modell dominante kutt

```
model NSB_bender
uses "mmxprs"
uses "mmsystem"

(!
Modell for å sjekke dominerende kutt

Marianne Risberg
Masteroppgave våren 2005 NTNU

!)



starttid := gettime
declarations
LOVLIG: integer
TID: real
TYPE: set of integer
UKE: integer
NODER: set of integer
DAGSLOPSNODER: set of integer
ANTALLNODER: integer
ANTALLDAGSLOP: integer
ANTALLVLHNODER: integer
ADINNODER: array(NODER,TYPE) of integer
DINNODER: dynamic array(NODER,NODER,TYPE) of integer
AVUTNODER: array(NODER,TYPE) of integer
VUTNODER: dynamic array(NODER,NODER,TYPE) of integer
AVINNODER: array(NODER,TYPE) of integer
VINNODER: dynamic array(NODER,NODER,TYPE) of integer
M1: dynamic array(NODER,NODER,TYPE)of real
M2:dynamic array(NODER,NODER) of real
M3: dynamic array(NODER,NODER) of real
KMTT: dynamic array(NODER,NODER) of real
LENGDE: array(DAGSLOPSNODER) of real
KMGRENSE: array(TYPE) of real

end-declarations

initializations from "generert_graf.dat"
UKE
TYPE
```

```
NODER
DAGSLOPSNODER
ANTALLNODER
ANTALLDAGSLOP
ANTALLVLHNODER
AVUTNODER
VUTNODER
DINNODER
ADINNODER
VINNODER
AVINNODER
M1
M2
M3
KMTT
LENGDE
KMGRENSE
end-initializations

initializations from "lovlig.dat"
LOVLIG
end-initializations
!LOVLIG := 20

declarations

dualE1: array(NODER,NODER,TYPE, 1..LOVLIG) of real
dualE2: array(NODER,NODER,TYPE, 1..LOVLIG) of real
dualE3: array(NODER,NODER,TYPE, 1..LOVLIG) of real
dualE4: array(NODER,TYPE, 1..LOVLIG) of real


AKTIV: array(TYPE, 1..UKE, 1..LOVLIG) of integer


y: array(TYPE, 1..UKE, 1..LOVLIG) of mpvar
dom: array(TYPE,1..UKE,1..LOVLIG,1..LOVLIG) of mpvar

end-declarations

initializations from "losning_subproblemstraale.dat"
dualE1
dualE2
dualE3
dualE4
end-initializations


initializations from "aktiv.dat"
AKTIV
end-initializations



forall(k in TYPE, u in 1..UKE, l1 in 1..LOVLIG|AKTIV(k,u,l1)=0) do
```

```
create (y(k,u,l1))

forall(l2 in 1..LOVLIG|l2<>l1) create(dom(k,u,l1,l2))
forall(l2 in 1..LOVLIG|l2<>l1) dom(k,u,l1,l2) is_binary


forall(j in NODER|j<=ANTALLDAGSLOP, a in 1..ADINNODER(j,k), i in DAGSLOPSNODER|
              i=DINNODER(j,a,k))do

floor(M1(DINNODER(j+ANTALLNODER*(u-1),a,k),j+ANTALLNODER*(u-1),k)*
dualE1(DINNODER(j+ANTALLNODER*(u-1),a,k),j+ANTALLNODER*(u-1),k,l1))*y(k,u,l1) >=

SUM(l2 in 1..LOVLIG|l2<>l1 and AKTIV(k,u,l2)=0 and l1 <=30 and l2<=30)
(dom(k,u,l1,l2)*floor(M1(DINNODER(j+ANTALLNODER*(u-1),a,k),j+ANTALLNODER*(u-1),k)*
dualE1(DINNODER(j+ANTALLNODER*(u-1),a,k),j+ANTALLNODER*(u-1),k,l2)))

end-do


forall(j in NODER|j>ANTALLDAGSLOP and j<=ANTALLNODER, a in 1..AVINNODER(j,k))do

floor(M2(VINNODER(j+ANTALLNODER*(u-1),a,k),j+ANTALLNODER*(u-1))*
dualE2(VINNODER(j+ANTALLNODER*(u-1),a,k),j+ANTALLNODER*(u-1),k,l1))*y(k,u,l1)  >=

SUM(l2 in 1..LOVLIG|l2<>l1 and AKTIV(k,u,l2)=0 and l1 <=30 and l2<=30)
(dom(k,u,l1,l2)*floor(M2(VINNODER(j+ANTALLNODER*(u-1),a,k),j+ANTALLNODER*(u-1))*
dualE2(VINNODER(j+ANTALLNODER*(u-1),a,k),j+ANTALLNODER*(u-1),k,l2)))
end-do


forall(j in NODER|j<ANTALLDAGSLOP and j<=ANTALLNODER, a in 1..AVUTNODER(j,k)) do
floor(M3(j+ANTALLNODER*(u-1),VUTNODER(j+ANTALLNODER*(u-1),a,k))*
dualE3(j+ANTALLNODER*(u-1),VUTNODER(j+ANTALLNODER*(u-1),a,k),k,l1))*y(k,u,l1) >=

SUM(l2 in 1..LOVLIG|l2<>l1 and AKTIV(k,u,l2)=0 and l1 <=30 and l2<=30)
dom(k,u,l1,l2)*floor(M3(j+ANTALLNODER*(u-1),VUTNODER(j+ANTALLNODER*(u-1),a,k))*
dualE3(j+ANTALLNODER*(u-1),VUTNODER(j+ANTALLNODER*(u-1),a,k),k,l2))
end-do


floor((
sum(j in NODER|j<=ANTALLDAGSLOP, a in 1..ADINNODER(j,k))
(M1(DINNODER(j+ANTALLNODER*(u-1),a,k),j+ANTALLNODER*(u-1),k)-
KMTT(DINNODER(j+ANTALLNODER*(u-1),a,k),j+ANTALLNODER*(u-1))-
LENGDE(j+ANTALLNODER*(u-1)))*
dualE1(DINNODER(j+ANTALLNODER*(u-1),a,k),j+ANTALLNODER*(u-1),k,l1) +

sum(j in NODER|j>ANTALLDAGSLOP and j <=ANTALLNODER, a in 1..AVINNODER(j,k))
(M2(VINNODER(j+ANTALLNODER*(u-1),a,k),j+ANTALLNODER*(u-1))-
KMTT(VINNODER(j+ANTALLNODER*(u-1),a,k),j+ANTALLNODER*(u-1))+KMGRENSE(k))*
dualE2(VINNODER(j+ANTALLNODER*(u-1),a,k),j+ANTALLNODER*(u-1),k,l1)+

sum(j in NODER|j>ANTALLDAGSLOP and j<=ANTALLNODER, a in 1..AVUTNODER(j,k))
(M3(j+ANTALLNODER*(u-1),VUTNODER(j+ANTALLNODER*(u-1),a,k))-
```

```
KMTT(j+ANTALLNODER*(u-1),VUTNODER(j+ANTALLNODER*(u-1),a,k))-
LENGDE(VUTNODER(j+ANTALLNODER*(u-1),a,k)))*
dualE3(j+ANTALLNODER*(u-1),VUTNODER(j+ANTALLNODER*(u-1),a,k),k,l1)+

sum(j in NODER|j<=ANTALLDAGSLOP)dualE4(j+ANTALLNODER*(u-1),k,l1)))*y(k,u,l1)<=

 sum(l2 in 1..LOVLIG|l1<>l2 and AKTIV(k,u,l2)=0)dom(k,u,l1,l2)* floor(
(sum(j in NODER|j<=ANTALLDAGSLOP, a in 1..ADINNODER(j,k))
(M1(DINNODER(j+ANTALLNODER*(u-1),a,k),j+ANTALLNODER*(u-1),k)-
KMTT(DINNODER(j+ANTALLNODER*(u-1),a,k),j+ANTALLNODER*(u-1))-
LENGDE(j+ANTALLNODER*(u-1)))*
dualE1(DINNODER(j+ANTALLNODER*(u-1),a,k),j+ANTALLNODER*(u-1),k,l2) +

 sum(j in NODER|j>ANTALLDAGSLOP and j <=ANTALLNODER, a in  1..AVINNODER(j,k))
(M2(VINNODER(j+ANTALLNODER*(u-1),a,k),j+ANTALLNODER*(u-1))-
KMTT(VINNODER(j+ANTALLNODER*(u-1),a,k),j+ANTALLNODER*(u-1))+
KMGRENSE(k))*dualE2(VINNODER(j+ANTALLNODER*(u-1),a,k),j+ANTALLNODER*(u-1),k,l2)+

 sum(j in NODER|j>ANTALLDAGSLOP and j<=ANTALLNODER, a in  1..AVUTNODER(j,k))
(M3(j+ANTALLNODER*(u-1),VUTNODER(j+ANTALLNODER*(u-1),a,k))-
KMTT(j+ANTALLNODER*(u-1),VUTNODER(j+ANTALLNODER*(u-1),a,k))-
LENGDE(VUTNODER(j+ANTALLNODER*(u-1),a,k)))*
dualE3(j+ANTALLNODER*(u-1),VUTNODER(j+ANTALLNODER*(u-1),a,k),k,l2)+
 sum(j in NODER|j<=ANTALLDAGSLOP)dualE4(j+ANTALLNODER*(u-1),k,l1) ))




Total := y(k,u,l1)- sum(l2 in 1..LOVLIG|l1<>l2)dom(k,u,l1,l2) !+ m(k,u,l1,l2)



maximize(Total)




problemstatus := getprobstat

if (problemstatus = XPRS_OPT) then

if(getsol(y(k,u,l1))>0 and sum(l2 in 1..LOVLIG|l1<>l2)getsol(dom(k,u,l1,l2))>0) then

AKTIV(k,u,l1):= 1

end-if

elif(problemstatus = XPRS_INF) then

AKTIV(k,u,l1) := 0
```

```
elif(problemstatus = XPRS_UNB)then

AKTIV(k,u,l1) := 0
else
writeln("enda en status til")
end-if


end-do


ANTALL := sum(k in TYPE, u in 1..UKE, l in 1..LOVLIG) AKTIV(k,u,l)
writeln("Antall aktive kutt: ", ANTALL)



initializations from "losningstid_kutt.dat"
TID
end-initializations

fopen("losningstid_kutt.dat", F_OUTPUT)
writeln("TID: ", gettime - starttid + TID)
fclose(F_OUTPUT)

initializations to "aktiv.dat"
AKTIV
end-initializations

end-model
```

Vedlegg E: Hensettingskapasitet

I dette vedlegget presenteres resultater etter kjøring av formulering Xpress, der hensettingskapasitet er tatt hensyn til og modellerert som vist i kapittel 4.1. Her blir tomtogsbehovet identifisert og bestemt om det skal kjøre på morgenen eller på kvelden. Hvis man studerer antallet variabler og restriksjoner for dette problemet kontra hvis hensettingskapasitet ikke er tatt med i tabell E.1, ser man at problemene er betydelig større i antall variabler og antall restriksjoner med hensettingskapasitet.

Tabell E.1: Resultater med hensettingskapasitet i Xpress

Data-	Med hensetting		Uten hensetting		Løse-	TT	VLH	Stas-
sett	Variabel[a]	Restr.[a]	Variabel[a]	Restr.[a]	tid	km		joner
typer1	1250/944	1371/995	120/95	157/126	4.6s	325.1	1	11
typer2	2968/2342	3103/2434	276/271	327/324	4.4s	768.4	3	11
typer3	2860/2240	2995/2332	219/216	274/271	4.2s	13.2	3	11
typer4	2896/2274	3031/2366	270/267	325/322	12.9s	478.4	3	11
typer7	5490/4298	5639/4412	499/493	562/560	87.2s	710.5	4	11
typer8	8576/6654	8739/6782	769/763	846/844	20758.0s	1109.4	5	11
typer692[b]	102535/84235	102935/84488	4038/3922	4129/4037	13343.1s	995.0	2	25
typer72	36465/28524	36733/28778	3131/3081	3262/3260	62336.4s**	3814.3	10	11

a Formulert/presolved

b Har satt den nedre grensen til 2 vedlikehold.

** Ikke løst til optimalitet.

Problemstørrelsen er avhengig av antall stasjoner. For typer692 er det mange flere stasjoner enn for de andre datasettene. Dette skyldes at datasettene har blitt generert etter hverandre, og det andre datasettet er kun en utvidelse av det første og så videre. Hvis man ser på problemet for typer 72, har jeg kun generert en liste med stasjoner som dagsløpene starter og stopper på. Hvis flere stasjoner skal inkluderes, økes problemet tilsvarende. For typer 72 er det en liste på 11 stasjoner. Datasettene med togtype 69-3 og 69-3 er generert etter datasettene med togtype 72, så jeg har valgt å bruke stasjonene fra type 72 som basis, og inkludert alle de andre stasjonene som dagsløpene fra type 69 starter eller stopper på. Derfor er antallet stasjoner større for typer692 enn for typer72. For typer692 er det en liste med 25 stasjoner, og dette gjør at typer692 blir et mye større problem i antall variable og restriksjoner.

Antall tomtogskilometer kan også variere noe når man kjører modellen med hensettingskapasitet. *KMTT*-tabellen er ikke generert helt riktig. Mange av avstandene er beregnet litt tilfeldig, og det vil derfor ikke nødvendigvis gi samme avstand å kjøre direkte mellom to stasjoner som å dele opp strekningen i to slik som blir gjort i problemet som hensyntar hensettingskapasitet.