Vedlikeholdsplanlegging for NSB

Janne Bjorvand og Marianne Risberg

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

13. desember 2004

Forord

Denne oppgaven er en del av fordypningsemnet i 5. årskurs på sivilingeniørstudiet ved NTNU. Fordypningsemnet består av teoritema og et skriftlig arbeid, og denne rapporten utgjør det skriftlige arbeidet. Oppgaven er skrevet ved Institutt for industriell økonomi og teknologiledelse innenfor fordypningsretningen Anvendt økonomi og optimering.

Vi ønsker å rette en takk til NSB Persontog Plan og veilederne våre Tore Tomasgard og Kenneth Aschehoug ved NSB for en interessant oppgave og hjelp underveis i prosjektet. Vi ønsker også å takke Marte Fodstad på SINTEF for god hjelp i oppstartsfasen og veilederen vår Asgeir Tomasgard for veiledning underveis. Tilslutt vil vi rette en takk til Bjørn Nygreen for god hjelp med problemforståelse og modellering.

Janne Bjorvand og Marianne Risberg

Trondheim, 13.desember, 2004

Sammendrag

I denne oppgaven har vi sett på planlegging av driftspausebasert vedlikehold i NSB. Dette vedlikeholdet blir gjennomført i pauser i materiellturneringen. Avgjørende faktorer for planlegging av vedlikehold kan være antallet, kapasiteten og den geografiske plasseringen til vedlikeholdsbasene, frekvensen på vedlikeholdet og reservebeholdningen av materiell.

I litteraturen om materiellplanlegging har vi i hovedsak sett to angrepsmåter til planlegging av vedlikehold, tidsavhengig eller distanseavhengig. I NSB er vedlikeholdet avhengig av distanse. For hver togtype har NSB en kilometergrense som gir maksimalt antall kilometer som er lovlige å kjøre mellom to vedlikehold.

Materiellplanleggingen i NSB forgår ved at en først tilordner materiell til en gitt ruteplan og deretter planlegger vedlikeholdet. NSB har i dag ikke et beslutningsstøtteverktøy for å planlegge dette vedlikeholdet. Arbeidet i dette prosjektet er et første forsøk på å modellere og implementere en slik beslutningsstøtte. Vi har tatt for oss lokaltogene på Østlandet, og ser på planlegging på taktisk nivå.

Inndataene til problemet består av dagsløp og mulige vedlikehold. Alle turene i ruteplanen finnes i et og bare et dagsløp, og alle dagsløpene må dekkes av en bestemt togtype. Hovedmålet til modellen er å minimere antall vedlikehold ved å knytte sammen dagsløpene slik at kravet til vedlikehold blir tilfredsstilt.

Vi har modellert vedlikeholdsproblemet ved hjelp av en nettverksformulering. Egenskapene til dette nettverket er komplisert. Vi har derfor kun klart å løse vedlikeholdsproblemet for små datasett. Ved testing av større datasett har vi kun funnet lovlig løsninger, og for det totale problemet har vi ikke funnet noen løsning. Vi har imidlertid vist at det kan være mulig å redusere antall vedlikehold ved bruk av optimeringsverktøy.

I vår implementering har vi brukt Xpress som har Branch and Bound som løsningsstrategi. For å kunne løse det komplette vedlikeholdsproblemet må det dermed brukes en mer effektiv løsningsalgoritme.

Innhold

Figurer

Tabeller

Kapittel 1

Innledning

I denne oppgaven ser vi på materiellplanlegging og spesielt planlegging av driftspausebasert vedlikehold for tog. Vi samarbeider med NSB, og problemformuleringen er laget og bestemt sammen med NSB.

1.1 Motivasjon og problemstilling

Det er ulik oppfatning om hvor viktig det er å planlegge driftspausebasert vedlikehold og i hvilken fase dette skal gjøres. Viktigheten av å planlegge vedlikeholdet er avhengig av:

Plassering av vedlikeholdsbaser. Hvis de fleste turene 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, vil dette kreve en nøyere planlegging enn om vedlikeholdet gjennomføres sjeldnere [Cordeau et al., 2001b].

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

Kapasitet på vedlikeholdsbasen. Hvis kapasiteten på en vedlikeholdsbase er lav, gir det mindre fleksibilitet og krever bedre planlegging.

Reserve. Hvis man har en liten reservebeholdning av materiell, må man planlegge vedlikehold nøyere. Dette skyldes at store deler av materiellparken er i drift [Tomasgard, 2004a, Erlebach et al., 2004].

Vedlikeholdsplanleggingen er viktig i NSB fordi vedlikeholdsbasene ikke er strategisk plassert. I dag ligger vedlikeholdsbasene der de historisk ble plassert, med utgangspunkt i enkeltstrekninger. Hvis man i dag kunne velge plassering for disse vedlikeholdsbasene, ville lokaliseringen høyst sannsynligvis vært annerledes. I NSB er frekvensen på

vedlikehold avhengig av en gitt kilometergrense per materielltype. Både antallet vedlikeholdsbaser og kapasiteten på disse er en begrenset ressurs, og generelt har NSB en liten reservebeholdning av materiell. Dette viser at behovet for taktisk planlegging av vedlikehold er til stede i NSB.

Målet for denne prosjektoppgaven er derfor å sette opp en modell for vedlikeholdsplanlegging for NSB. I samarbeid med veileder og NSB Plan velger vi å se på taktisk vedlikeholdsplanlegging av lokaltrafikken i Oslo.

1.2 Inndeling av oppgaven

I kapittel 2 presenteres caset som denne oppgaven bygger på. Vi gir en kort beskrivelse av dagens planleggingsprosess i NSB, og av hvilke betingelser som gjelder for materiellplanlegging.

Kapittel 3 inneholder operasjonsanalytisk teori. Dette kapitlet er tatt med for å gi et bedre grunnlag for å forstå operasjonsanalytiske begreper senere i rapporten, og som grunnlag for videre arbeid i en diplomoppgave.

Ulike problemstillinger for materiellplanlegging presenteres i kapittel 4. Her ser vi på ulike problemer med tilordning av materiell til en ruteplan og hvordan disse håndterer vedlikehold.

I Kapittel 5 avgrenser vi materiellplanleggingsproblemet fra kapittel 2 til vedlikeholdsproblemet vårt. Dette problemet modellerer vi ved hjelp av en nettverkformulering. Vi presenterer i tillegg et forslag til en modellformulering ved hjelp av set partitioning.

I kapittel 6 viser vi hvordan vi har implementert modellene og hvilke resultater vi fikk etter testing. Vi gir også noen kommentarer på muligheter for videre arbeid med modellen.

I vedlegg A finner man en ordliste som forklarer noe av togterminologien. I vedlegg B er koden som er skrevet i Xpress. I vedlegg C ligger et eksempel på inndata. Vedlagt på cd ligger koden, flere datasett og noen løsninger fra testing, samt bibliografien.

Kapittel 2

Case beskrivelse

I dette kapitlet vil vi presentere NSB og beskrive hvordan materiell og vedlikeholdsplanlegging foregår i dag.[1]

2.1 NSB Plan

NSB Persontog er delt opp i avdelingene Drift, Plan og Materiell samt Økonomi, Salg og Marked. Oppgavebeskrivelsen vår er utarbeidet i samarbeid med Planavdelingen.

NSB Plan står for den strategiske og taktiske planleggingen for NSB Persontog. Her lages produksjonsplaner som ruteplaner, materiellplaner og personellplan for lokførere i tillegg til vedlikeholdsplaner.

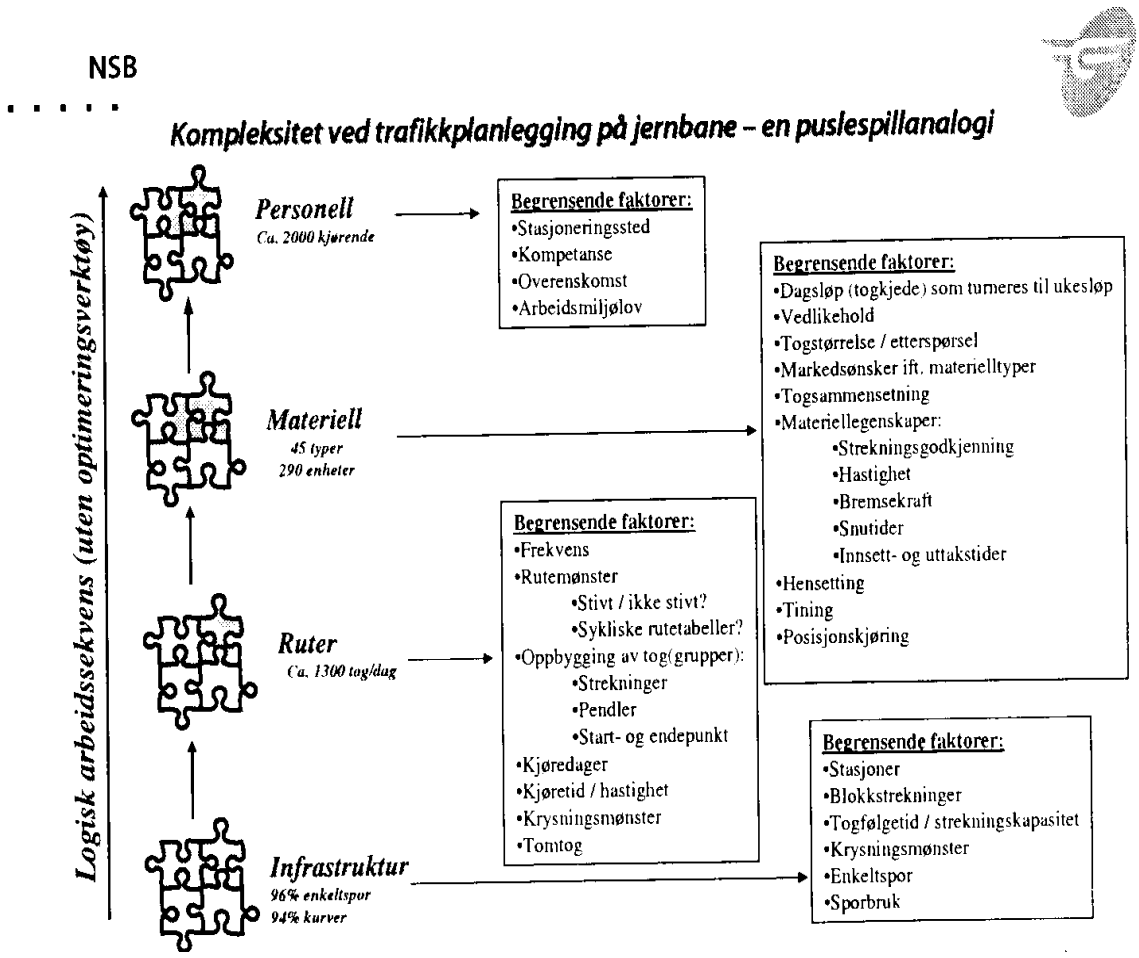
Planavdelingens hovedoppgave, med hensyn på materiell, er å planlegge alle bevegelser med NSBs persontogmateriell i langsiktige produksjonsplaner. Det skal sikres at det er tilstrekkelig med driftspauser til å utføre alle (materiellrelaterte) aktiviteter som er nødvendige for at NSBs togproduksjon skal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

2.2 Overordnet planleggingsprosess i NSB

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 som regel 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 praksis i større eller mindre grad de

[1] Denne beskrivelsen bygger på Aschehoug and Fodstad [2002], samtaler med veileder i NSB og ut fra tidligere sommerjobb i NSB.



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

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 et år før planen skal tre i kraft. Denne planleggingen bør ikke være for detaljert fordi sjansen er stor for at endringer vil oppstå. Den taktiske delen av planleggingsarbeidet skjer fra et år før planen skal realiseres og frem til hovedbestillingen sendes. Dette skjer 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 produksjonsalternativ. På dette nivået vet en hvilke avganger som skal kjøres og disse brukes som grunnlag for utarbeidelse av personellplaner, materiellturneringsplaner og vedlikeholdsplaner. Taktisk og detaljplanlegging blir ofte sett under ett, noe vi også vil gjøre 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 utarbeidete planene og deres realisering. Ved avvik er målet for NSB å justere seg tilbake til de opprinnelige taktiske planene.

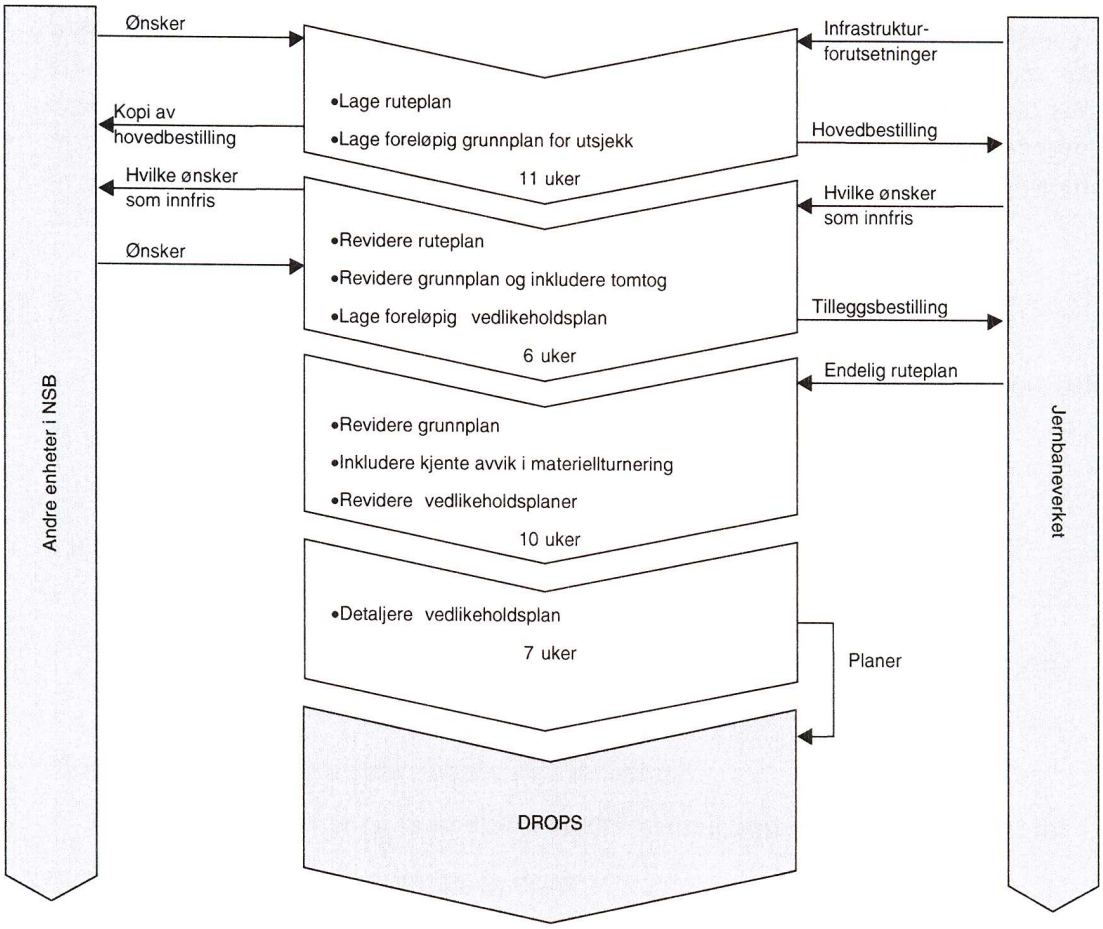
Det er to generelle trinn i planleggingsarbeidet som spesielt går igjen i materiell- og personellplanleggingen. Det første består i at det utarbeides en generell prinsipiell plan, uten at det er definert hvilket materiellindivid eller enkeltperson som skal utføre arbeidet. Neste trinn består i en individallokering, der det bestemmes hvilke individer som skal utføre hvilke planer [Olsson et al., 2002].

2.2.1 Materiell og vedlikeholdsplanlegging

Materiell og vedlikeholdsplanene utarbeides i hovedsak på det taktiske nivået. En framstilling av planprosessen med fokus på materiellplanlegging på taktisk nivå vises i figur 2.2.

Det første trinnet i materiellplanleggingen er å utarbeide en materiellturnering ut fra en ønsket ruteplan. Dette arbeidet vil derfor bestå i å fastsette en ruteplan, og å bestemme hvilken materielltype som skal dekke hver tur. Dette gjøres ved at turer knyttes sammen til sekvenser. Hver tur må bli dekket av en sekvens. Hver slik sekvens på en dag kalles et dagsløp. Hvert dagsløp har en lengde gitt i antall kilometer. Det blir laget dagsløp for hver dag i en uke. Alle disse dagsløpene utgjør en turnering. For at denne turneringen skal være lovlig må vedlikehold inkluderes. En må sjekke når det er tid til vedlikehold, og dette kan være inni i et dagsløp eller mellom to dagsløp. Dagsløpene settes dermed slik at ingen sekvens av dagsløp overstiger den tillatte kjørelengden. Neste trinn i planleggingen vil så være å allokere materiell til turneringen.

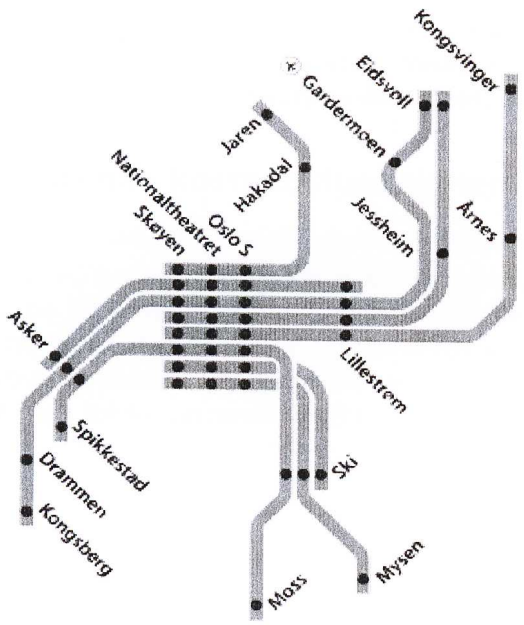
Det eksiterer ikke verken beslutningsstøtteverktøy for planleggingsprosessen som helhet [Sætermo and Tomasgard, 2001] eller for materiellplanleggingen. Planleggingsprosessen er i dag tidkrevende, og man rekker ikke å gjennomføre mer enn en iterasjon på halvårsplanene.



Figur 2.2: Planprosessen med fokus på materiell.
DROPS og Jernbaneverket er enheter utenfor planavdelingen [Aschehoug and Lauritzen, 2002].

2.3 Vedlikehold hos NSB

For lokaltogene planlegger NSB vedlikehold etter at materiell for hver tur er bestemt og etter at turene er satt sammen til dagsløp. Togene må vedlikeholdes innenfor bestemte kilometerintervaller, og denne grensen er forskjellig for hver materielltype.



Figur 2.3: Lokaltogene på Østlandet.
[Aschehoug and Lauritzen, 2002].

Lokaltogstrafikken på Østlandet har flere materielltyper og vedlikeholdsbaser. En oversikt over strekningene finnes i figur 2.3. Regionen har to vedlikeholdsbaser, Sundland ved Skøyen og Filipstad ved Drammen. Materiellparken består av to materielltyper, type 72 og type 69. Type 69 består av to subtyper, 2-vognssett (69-2) og 3-vognssett (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.

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. Det er bare for type 69 de har en reserve på ca 10%. For de andre materielltypene er reservebeholdningen på 2-5% [Tomasgard, 2004a].

I lokaltogstrafikken 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[2].

[2]Grunnplanen fra januar 2005 til juni 2005

2.4 Generell problemstilling for materiellplanlegging i NSB

I dette kapitlet vil vi gi en kort oppsummering av det generelle materiellplanleggingsproblemet til NSB. For en mer utfyllende beskrivelse viser vi til Aschehoug and Fodstad [2002]. Materiellplanleggingen starter med å fastsette en ruteplan. Denne ruteplanen inneholder alle turene, med avgangs- og ankomsttider og avgangs- og ankomststeder, som skal kjøres. Deretter skal materiellplanleggeren tildele materiell til disse turene. Dette gjøres ved å sette opp gyldige turneringer. Setekapasiteten som hver tur skal dekke avgjøres hovedsaklig av markedsenheten i NSB.

2.4.1 Restriksjoner

Ved materiellplanlegging er det en del restriksjoner som det må tas hensyn til [Aschehoug and Fodstad, 2002].

Vedlikehold. For hver materielltype 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, derav navnet driftpausebasert vedlikehold. I tillegg må det være ledig tid på vedlikeholdsbasen for å få gjennomført vedlikeholdet.

Materielltilgjengelighet. NSB har et gitt antall tog av hver materielltype. Materiellplanleggeren må passe på at denne kapasiteten ikke blir overskredet.

Toglengde. Det er en maksimalgrense for hvor langt et tog kan være. Denne grensen avhenger av lengden på perrongene, effektiviteten ved stasjonsopphold for hver tur og av tekniske begrensninger.

Skjøting og deling. Det er kun tillatt å sette samme turer som inneholder tilstrekkelig tid til skjøting og deling. I tillegg må infrastrukturen på stasjonene tillate å gjennomføre dette.

Snutid. Hver sekvens må inneholde tilstrekkelig tid til snuing. 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. Dersom et tog må flyttes på grunn av disse kapasitetsbeskrakningene, settes det opp tomtog.

Sportilgang ved tomtogskjøring. Ved tomtogskjøring må det sjekkes om det er ledig sportilgang innenfor det gitte tidsintervallet for denne kjøringen.

Dekke etterspørselen på alle turene. Turene skal i så stor grad som mulig dekke etterspørselen fra markedet. Etterspørselen er gitt som antall togsett per tur.

Tining. For å kunne kjøre rutetilbudet på vinteren må togsettene inn til tining ved behov.

2.4.2 Mål

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

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.

Sykliske turneringer. Det er ønskelig at hvert tog etter en hvis tid kommer tilbake til utgangspunktet og derfra gjentar turneringen. Dette øker robustheten. Av samme grunn er det derfor også ønskelig at syklene skal være så korte som mulig.

Personellkostnader. Tomtogskjøring, skjøting og deling, parkering og tilsyn av materiell krever personell. Materiellplaneleggeren ønsker disse personellkostnadene så lave som mulig.

Vedlikeholdskostnader. Vedlikeholdskostnadene avhenger av antall kilometermeter som blir tilbakelagt. Matriellplanleggeren vil at disse kostnadene skal være så lave som mulig.

Kapitalkostnader. Den største kapitalkostnaden har man ved kapitalbinding i togmateriellet. Derfor er det ønskelig med en materiellturnering som krever lite materiell.

Lik kjørelengde. Det er ønskelig at hvert sett i gjennomsnitt har kjørt like langt slik at slitasjen på settene er lik.

LIFO ved stasjonene. På noen stasjoner er det begrenset med hensettingskapasitet. Dersom flere togsett settes på samme spor, er det ønskelig at det toget det som kommer sist inn på kvelden kjører først ut igjen på morgenen (LIFO - last inn first out).

Kapittel 3

Operasjonsanalytisk teori

I dette kapitlet vil vi forklare noen ulike operasjonsanalytiske modeller, og noen teknikker for å løse større modeller.

Det som blir presentert i dette kapitlet er momenter som vi har kommet over når vi har lest om materiellplanleggingsproblemet.

3.1 Nettverksmodeller

Nettverksmodeller har en struktur som kan vises i en *rettet graf*. Denne grafen består en et sett med *noder*, V, og et sett med *kanter*, A. En node en enten en *tilbyder*node, en *etterspørsels*node eller en *transport*node. Hver kant blir gitt ved de to nodene den binder sammen. Det vil si om at en kant binder sammen node i og node j vil kanten bli gitt ved (i,j). At en kant er rettet betyr at retningen på flyten over kanten har betydning. For kant (i,j) vil retningen på flyten være fra node i til node j. Beslutningsvariablen x_{ij} i en nettverksmodell sier hvor mye flyt det er over kant (i,j).

Standardform for et nettverksproblem med rettet graf med noder $k \in V$, etterspørsel b_k, kanter $(i,j) \in A$, kapasitet $u_{i,j}$ og enhetskostnad $c_{i,j}$:

$$min \sum_{(i,j)\in A} c_{i,j}x_{i,j} \tag{3.1}$$

$$\sum_{(i,k)\in A} x_{i,k} - \sum_{(k,j)\in A} x_{k,j} = b_k \qquad \forall k \in V \tag{3.2}$$

$$0 \leq x_{i,j} \leq u_{i,j} \qquad (i,j) \in A \tag{3.3}$$

(3.1) er målfunskjonen som minimerer kostnadene. (3.2) er hovedbeskrankningen i et nettverksproblem som sørger for balanse i hver node. Balanse i en node vil si flyten inn i noden minus flyten ut av noden er lik netto etterspørsel for denne noden. I en slik modell vil summen av flyt ut fra tilbyder nodene være lik flyten inn i etterspørselsnodene. Vi har dermed ikke overskudd eller underskudd av flyt. (3.3) er kapasitetsbeskrankningen for hver hver kant [Rardin, 1998].

3.1.1 Heltallsegenskapen til nettverksmodeller

Hvis en minimum nettverksmodell med heltallsverdier i alle beskrakningene har en optimal løsning, vil denne løsningen være heltallig. Dette betyr at modellen har heltallsverdier for alle tilbyder og etterspørsels nodene, samt heltallige flytkapasiteter. Denne heltallsegenskapen er ikke nødvendig for kostnadene i målfunskjonen [Rardin, 1998].

3.1.2 Transportproblem

Et transportproblem er et spesielt minimum nettverksflytproblem der hver node er enten bare en tilbydernode eller bare en etterspørselsnode.

Standard form for transportproblem med tilbyder s_i, etterspørsel d_j og kostnader c_{ij}:

$$Min \sum_i \sum_j C_{ij} x_{ij} \tag{3.4}$$

$$\sum_j x_{ij} = s_i \qquad \forall i \tag{3.5}$$

$$\sum_i x_{ji} = dj \qquad \forall j \tag{3.6}$$

$$x_{ij} \geq 0 \qquad \forall i, j \tag{3.7}$$

(3.4) er målfunskjonen som minimerer transportkostnadene. (3.5) gir flyt ut for hver tilbydernode og (3.6) gir flyt inn for hver etterspørselsnode. (3.7) er ikke-negativitetskravet til bestlutningsvariablene [Rardin, 1998].

3.1.3 Tilordningsproblem

Tilordningsproblemer omhandler optimal tilpasning av objekter (noder) i to atskilte sett.

Et eksempel på et tilordningsproblem er når man for har n skip og n laster som skal hentes av skip, der hvert skip bare kan hente en last og hver last må hentes av bare et skip. I dette problemet vil beslutningsvariabelen, x_{ij}, være binær og lik en hvis skip i skal hente last j og null ellers. Målfunksjonen kan være å minimere avstanden som skipene samlet må kjøre for å hente lasten, gitt at ikke alle skipene er plassert på samme sted i utgangspunktet.

Standard form for et lineært tilordningsproblem med kostnadsmatrise C_{ij} :

$$Min \sum_{i \in N} \sum_{j \in N} C_{ij} x_{ij} \tag{3.8}$$

$$\sum_{i \in M} x_{ij} = 1 \qquad \forall j \in N \tag{3.9}$$

$$\sum_{j \in N} x_{ji} = 1 \qquad \forall i \in M \tag{3.10}$$

$$x_{ij} \in \{0,1\} \qquad \forall i \in M, j \in N \tag{3.11}$$

(3.8) er målfunksjonen som minimerer kostnaden ved transporten. (3.9) og (3.10) sørger henholdvis for at et objekt $j \in M$ blir tilordnet et objekt $i \in N$, og at et objekt $i \in N$ blir tilordnet et objekt $j \in M$, $M = N$. (3.11) er binærkravet til beslutningsvariablene. Summene er begrenset til de kantene (i, j) som er tillatt i problemet.

Alle lineære tilordningsproblemer kan formuleres som et nettverksflytproblem. Alle objektene vil være noder i nettverket, der det vil gå en kant fra hver node i det ene settet til alle nodene i det andre settet. Flyt ut av hver node i det ene settet (tilbydernoder) vil være lik en flyt inn i hver node i det andre settet (etterspørselsnoder). Denne flyten har verdi lik 1. Når man formulerer problemet på denne måten kan beslutningsvariabelen formuleres som en kontinuerlig variabel, det vil si at binærkravet blir relaksert til $0 \leq x_{i,j} \leq 1$. Heltallsegenskapen for nettverksmodeller, se (3.1.1), vil sørge for at en optimal løsning i det linære problemet vil være en heltallsløsning, det vil her si binære verdier [Rardin, 1998].

I stedet for å si at hver node j må tilordnes akkurat en node i og motsatt, kan en tillate at j kan bli tilordnet flere noder i. Dette vil gi et *generelt tilordningsproblem*.

Standardform for et generelt tilordningsproblem der b_j er kapasiteten til j, og s_{ij} er mengden av kapasiteten som blir konsumert dersom det er flyt over (i, j):

$$Min \sum_{i \in N} \sum_{j \in N} C_{ij} x_{ij} \tag{3.12}$$

$$\sum_{i \in M} s_{ij} x_{ij} \leq b_j \qquad \forall j \in N \tag{3.13}$$

$$\sum_{j \in N} x_{ji} = 1 \qquad \forall i \in M \tag{3.14}$$

$$x_{ij} \in \{0,1\} \qquad \forall i \in M, j \in N \tag{3.15}$$

Beskrankningene for dette generelle tilordningsproblemet vil være tilsvarende beskrankningene for det lineære problemet borsett fra (3.13). (3.13) er kapasitetsbeskrankningen til hver node j.

Både lineære og generelle tilordningsproblem ser på tilordning av noder i ulike sett. *Matching* problemer unngår dette ved å definere beslutningsvariabelen slik:

$x_{ii'}$ er lik en hvis i er koblet til i' og null ellers.

$$x_{ii'} = \begin{cases} 1 & \text{hvis i er koblet til i'} \\ 0 & \text{ellers} \end{cases}$$

Standard formulering av et matching problem:

$$Min \sum_{i \in N} \sum_{i' \in N} C_{ii'} x_{ii'} \tag{3.16}$$

$$\sum_{i' < i} x_{i'i} + \sum_{i' > i} x_{ii'} \qquad \forall i \; in M \tag{3.17}$$

$$x_{ii'} \in \{0,1\} \qquad \forall i \; in M, i' > i \tag{3.18}$$

Et matching problem er en spesiell utgave av set packing problemer. Det kommer mer om slike problemer i kapittel 3.2.

3.1.4 Flervareflyt problem

Transportproblemer og tilordningsproblemer er de mest kjent spesialtilfellene av nettverkflytsmodeller. Disse problemene kan karakteriseres som et *envareflytproblem* (*eng: single-commodity flow models*). Disse problemene kan bli modellert ved å betrakte hvor mye av en vare som entrer eller forlater nettverket i hver node og hvor mye som flyter gjennom hver kant [Rardin, 1998].

Flervareflytproblemer (*eng: multicommodity flow models*) oppstår når varene som beveger seg gjennom et nettverk må holdes separat. Varer som flyter gjennom en flervaremodell kan ikke bli analysert separat fordi de flyter over felles kanter som har en maksimal flytkapasitet. Flervareflytmodeller ønsker å minimere kostnadsflyt der separate varer forflytter seg gjennom et felles nettverk [Rardin, 1998].

Standardform for en flervareflyt modell i en rettet graf med noder $k \in V$, kanter $(i,j) \in A$, enhetskostnad $c_{q,i,j}$ for vare q i (i,j), kapasitet $u_{i,j}$ for kant (i,j), etterspørsel $b_{q,k}$ for vare q i node k og beslutningsvariabel $x_{q,i,j}$ for vare q over kant (i,j):

$$min \sum_{q} \sum_{(i,j)\in A} c_{q,i,j} x_{q,i,j} \tag{3.19}$$

$$\sum_{(i,k)\in A} x_{q,i,k} - \sum_{(k,j)\in A} x_{q,k,j} = b_{q,k} \qquad \forall q, k \in V \tag{3.20}$$

$$\sum_{q} x_{q,i,j} \leq u_{i,j} \qquad (i,j) \in A \tag{3.21}$$

$$x_{q,i,j} \geq 0 \qquad q, (i,j) \in A \tag{3.22}$$

(3.19)-(3.22) tilsvarer (3.1)-(3.3) bortsett fra den nye indeksen q som indikerer at vi har flere varer.

Heltallsegenskapen til envareflytproblem gjelder ikke for flervareflytproblemer [Rardin, 1998]. Dette betyr at selv om alle dataene i et flervareflytproblem er heltallige, så kan den optimale løsningen av et flervareflytproblem likevel bli fraksjonell.

Et eksempel på et flervareflytproblem får vi ved å utvide eksemplet fra kapittel 3.1.3 til å omhandle flere typer lasteskip. Disse skipene skal hente ulike varer i noen havner (noder), og frakte disse til andre havner i nettverket. For hver havn er det et maksimalt antall skip som kan legge til. Denne grensen vil bli gitt av en flytkapasitets beskrankning inn i hver havn. Denne beskrankningen vil være felles for lasteskipene og de ulike skipstypene kan derfor ikke modelleres separat. Samtidig vil hvert lasteskip ha en øvre kapasitet på hvor mye varer det kan frakte.

3.2 Set partitioning problem

Det er også enn annen måte å løse problemene på enn ved å formulere problemet som et nettverk og bruke bueflyt. Man kan formulere problemet med heltall. Først spesifiseres en sekvens av kolonner som representerer mulige riktige delløsninger til problemet. Man løser deretter en set partitioning modell (eventuelt set cover modell eller set packing modell) for å velge en optimal samling av disse kolonnene som oppfyller restriksjonene i problemet [Rardin, 1998].

Set partitioning, set cover og set packing er tre alternative modeller som går ut på å bestemme medlemskap i bestemte undersett. I set cover er man medlem i minst et subset (overdekning), mens set packing er maksimalt et sett (underdekning). Set partitioning er medlemskap i kun et subsett uten overlapp.

Det som kjennetegner slike problemer er at de har binære variable. Hvis man betrakter en nettverksmodell vil beslutningsvariabelen x_{ijv} være lik en hvis kjøretøy v kjører direkte fra i til j og lik null ellers og kan i mange tilfeller lineærrelakseres, se kapittel 3.1.3. Beslutningsvariabelen i et set partitioning problem er x_{rv} og lik en hvis kjøretøy v kjører rute r

Et eksempel på en set partitioning formulering:

$$\begin{aligned} min \qquad & \sum_i C_i x_i \\ \sum_i A_{ij} x_i = 1 \qquad & \forall j \\ x_i\{0,1\} \qquad & \forall i \end{aligned} \tag{3.23}$$

C_i er kostnaden til delmengde i, x_i er 1 hvis delmengde i er med i losningen og 0 ellers, og A_{ij} spesifiserer hvorvidt delmengde i dekker element j i mengden, og vil derfor være enten 0 eller 1. Hver kolonne i A_{ij} vil derfor være en delmengde. Problemet er å velge ut et sett med delmengder slik at de til sammen dekker nøyaktig hele mengden. Samtidig finner modellen det billigste settet med delmengder som dekker hele mengden nøyaktig.

Ved å endre (3.23) i set partitioning problemet kan man få set cover eller set packing problem:

$$\begin{array}{lll} \text{Set cover:} & \sum_i A_{ij} x_i \geq 1 & \forall j \\ \text{Set packing:} & \sum_i A_{ij} x_i \leq 1 & \forall j \end{array}$$

3.3 Lineær relaksering

Relakseringer av et problem brukes til å finne grenser for den optimale verdien for problemet. I et ILP (*eng: Integer Linear Program*) eller et MILP (*eng: Mixed-Integer Linear Program*) kan en eller flere av heltallsvariablene bli relaksert til kontinuerlige variable, mens resten av beskrankningene beholdes i sin opprinnelige form. Løsningen på det relakserte problemet vil derfor gi en nedre (øvre) grense for den optimale verdien i et minimeringsproblem (maksimeringsproblem). Hvis det relakserte problemet ikke har noen løsning, vil heller ikke det opprinnelige problemet ha noen løsning. I tillegg hvis

løsningen av det relakserte problemet også er gyldig i det opprinnelige problemet, vil det relakserte problemet gi en optimal løsning. Alt dette fordi løsningsrommet til det relakserte problemet alltid vil innholde minimum løsningsrommet til det opprinnelige problemet [Rardin, 1998].

3.4 Lagrange relaksering

Ved *Lagrange relaksering* blir, i motsetning til lineær relaksering, ikke heltallskravet droppet. Derimot blir en eller flere av de lineære beskrankningene relaksert. Dette skjer ved at de blir dualisert eller vektet inn i målfunksjonen med passende *Lagrangemultiplikatorer* for å unngå ulovligheter. Løsningen av dette relakserte problemet vil i likhet med lineær relaksering gi en nedre (øvre) grense på den optimale løsningen på det opprinnelige minimeringsproblemet (maksimeringsproblemet). Kvaliteten på den nedre grensen vil være avhenging av hvilke verdier som blir valgt på Lagrangemultiplikatorene, λ. Hvis den relakserte beskrankningen har formen $Ax \geq b$ vil man sette $\lambda \geq 0$ for et minimeringsproblem. Lagrangemultiplikatoren settes derimot $\lambda \leq 0$ dersom $Ax \leq b$. Motsatte verdier for λ vil velges for et maksimeringsproblem. Dersom $Ax = b$ vil fortegnet til λ være ubestemt. Dermed vil alltid bidraget fra den relakserte restriksjonen gjøre målfunksjonsverdien bedre, det vil si lavere (høyere) for et minimeringsproblem (maksimeringsproblem). To generelle teknikker for å velge verdier på disse multiplikatorene er *subgradient optimization* og *mulitplier adjustment*.

Standard oppsett for Lagrange relaksering [Beasley, 1993]:

Starter med den opprinnelige modellen (P):

$$\begin{aligned} min \quad & cx \\ & Ax \geq b \\ & Bx \geq d \\ & x \in (0,1) \end{aligned} \tag{3.24}$$

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

Den lagrange relakserte modellen $(P1)$ blir dermed:

$$\begin{aligned} min \quad & cx + \lambda(b - Ax) \\ & Bx \geq d \\ & x \in (0,1) \end{aligned} \tag{3.25}$$

Vi vet at det lovlige området for (3.25) er minst like stort mulighetsområdet for (3.24). Når $\lambda \geq 0$ og $b - Ax \leq 0$ vil dette gi et ikke-positivt bidrag til målfunksjonen. Derfor vil modell (3.25) gi en nedre grense på (3.24). Dette problemet kalles *Lagrangean lower bound program* (LLBP).

Det vi 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*,(P2):

$$max_{\lambda \geq 0} \left\{ \begin{array}{ll} min & cx + \lambda(b - Ax) \\ & s.tBx \geq d \\ & x \in (0,1) \end{array} \right\} \tag{3.26}$$

3.5 Branch and bound

Branch and bound brukes for å finne løsninger på heltallsproblemer. Metoden søker gjennom en sekvens av løsninger helt til den finner den optimale løsningen eller til den har vist at det ikke finnes noen løsning for problemet.

En delvis (*eng:partial)* løsning i Branch and Bound er en løsning der noen av beslutningsvariablene er fastsatt til heltallsverdier og noen er relaksert. En inkumbent (*eng:incumbent*) node vil være den beste mulige løsningen som er funnet så langt i algoritmen. En foreldrenode (*eng:parent*) vil være den noden som en node springer ut ifra.

Branch and bound metoden starter med å lineærrelaksere alle beslutningsvariablene. Dette startproblemet vil være rotnoden i treet. Som nevnt tidligere vil vi ha en optimal løsning hvis det lineærrelakserte problemet gir en heltallig løsning. I tillegg vil det opprinnelige problemet være uløselig dersom det relakserte problemet er uløselig. Svaret for hver delvise løsning i treet vil gi en nedre grense for den optimale løsningen i et minimeringsproblem. Hvis man runder alle de fraksjonelle variabelverdiene opp til nærmeste heltall i et tilsvarende problem, vil dette gi en øvre grense for den optimale løsningen. Dette forutsetter at alle målfunksjonskoeffisientene er positive og at dette er en lovlig løsning. Motsatt vil gjelde for et maksimeringsproblem.

Med utgangspunkt i rotnoden må en velge en strategi som bestemmer:

1. hvilken node, det vil si hvilken delvise løsning, man skal forgrene på,
2. hvilken variabel man skal forgreine på og
3. om man eventuelt skal gjøre noe mer enn å bare løse MILP-problem i hver node.

Strategien for hvordan en skal bestemme hvilken node en skal forgrene på kan for eksempel være *beste først* eller *dybde først*. Beste først vil velge den noden som har den beste verdien på foreldre noden. Dybde først vil velge den noden der flest beslutningsvariable er fastsatt. Som navnet sier vil en med dybde først bevege seg langt ned i treet og dermed tidlig prøve å finne en mulig løsning på problemet.

For LP-basert *Branch and Bound* vil en alltid velge å forgrene ved å fastsette en heltallsrelaksert variabel som har en fraksjonell verdi. Hvis det er flere slike variabler i en node, kan en strategi bestå i fastsette den variabelen som har en verdi nærmest et heltall.

Branch and Bound terminerer når enhver delvis løsning i treet enten er forgrenet eller terminert. Hvis inkumbent løsningen har verdi, vil denne være løsningen på det opprinnelige problemet, ellers vil problemet være uløselig.

Hovedpoenget med denne metoden er at den deler opp mulighetsområdet for det opprinnelige problemet i mindre deler slik at hvert delproblem blir enklere å løse.

3.6 Benders dekomponering

Benders dekomponeringsalgoritme 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 [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, P [Eksempel er hentet fra forelesning i TIØ8: Optimering av transport og prosessproduksjon, 2004]:

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

I dette problemet er y heltall og x kontinuerlig.

For en gitt verdi på y reduseres P til et lineært underproblem, P_x, med kun kontinuerlige beslutningsvariable:

$$\begin{aligned} max \qquad & z = c^T x \\ & Ax \leq b - g(y) \\ & x \geq 0 \end{aligned}$$

Dualen D_1 til P_x er også et lineært subproblem:

$$\begin{aligned} min \qquad & w = [b - g(y)]^T u \\ & A^T u \geq c \\ & u \geq 0 \end{aligned}$$

Siden P_x er et lineært problem, følger det av dualitetsteoremet at den optimale løsningen av D_1 også vil gi optimal løsning av problem P_x. Sammenhengen mellom løsningen av primal og dualproblemet er vist i tabell 3.1.

u^k er ekstremalpunkt k i problem D_1, der $k \in K$ og K er antall ekstremalpunkter.

Hvis vi omformulerer D_1 på denne måten (D_2):

$$w = min_{u^k}[b - g(y)]^T u^k$$

og vi kjenner alle hjørnepunktene kan vi skrive P på denne måten (P*):

$$\begin{aligned} max \qquad & z = f(y) + min_{u_k}[b - g(y)]^T u^k \quad , \qquad (k = 0, 1, ..., K) \\ & y \in Y \end{aligned}$$

Tabell 3.1: Sammenheng mellom primalløsning og mulige dualløsninger

Primal	**Dual**
Optimal	Optimal
Umulig	Umulig eller uendelig
Uendelig	Umulig

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

$$\begin{aligned} & max \quad z \\ z \leq f(y) + [b - g(y)]^T u^k \quad & , \qquad (k = 0, 1, ..., K) \\ & y \in Y \end{aligned}$$

P^r vil ha bedre målfunksjonsverdi enn P^* fordi P^r har færre restriksjoner og dermed et større mulighetsområde.

En løsning av P^r gir z^{r+1}, y^{r+1} der $z^{r+1} \geq z^*$, mens en løsning av D_2 gir $f(y^r) + w^r \leq z^*$ der r er stegtelleren.

Da vil man finne verdier på begge sider av den optimale verdien til problemet. Når løsningene av P^r og D_2 er like, har man en optimal løsning. Hvis problemet P_x er uløselig vil heller ikke problem P ha noen løsning. Hvis D_2 er uløselig, vil problem P også være uløselig eller ikke ha noen endelig løsning.

3.7 Kolonnegenerering

Grunnideen til kolonnegenerering er å bare lage et lite subsett av kolonnene som inneholder optimal basis, og ignorere alle andre kolonner [Löbel, 1998]. For set partitioning problemer kan settet med alle kolonnene være veldig stort, og det vil da ta lang tid å generere alle kolonnene, i tillegg til at det blir et stort problem å løse.

Dantzig-Wolfe er en form for kolonnegenerering.

3.7.1 Dantzig-Wolfe dekomponering

Teorien i dette kapitlet er basert på Williams [1999] og Nygreen [1973]. Dantzig-Wolfe dekomponering kan brukes i problemer der en eller flere av beskrankningene kan deles inn i disjunkte mengder. Det kan for eksempel bety at vi kan dele restriksjonssettet $B_{ij}x_j = d_j$ i to separable restriksjonssett $B'_{ij}x_j = d'_j$ og $B''_{ij}x_j = d''_j$. Her vil B'_{ij} og B''_{ij} være en disjunkt oppdeling av B_{ij}, og d'_j og d''_j vil være en disjunkt oppdeling av d_j.

Da vil det opprinnelige problemet (DW):

$$
\begin{aligned}
min \qquad & \sum_j C_j x_j \\
\sum_j A_{ij} x_j = b_i \qquad & \forall i \in (1,k) \\
\sum_j B_{ij} x_j = d_i \qquad & \forall i \in (k+1,n) \\
x_j \geq 0 \qquad & \forall j
\end{aligned}
\tag{3.27}
$$

kunne omskrives til problemet (DW2):

$$
\begin{aligned}
min \qquad & \sum_j C_j x_j \\
\sum_j A_{ij} x_j = b_i \qquad & \forall i \in (1,k) && (3.28) \\
\sum_{j \in S_1} B'_{ij} x_j = d'_i \qquad & \forall i \in (k+1,m) && (3.29) \\
\sum_{j \in S_2} B''_{ij} x_j = d''_i \qquad & \forall i \in (m+1,n) && (3.30) \\
x_j \geq 0 \qquad & \forall j && (3.31)
\end{aligned}
$$

Mengdene S_1 og S_2 er de to separate mengdene som beslutningsvariabelen x_j kan deles inn i. Restriksjon (3.28) kalles her for *fellesbeskrankningen*. Denne restriksjonen kan her betraktes som en begrenset ressurs som både blir benyttet i ligningssettet (3.29) og (3.30).

Vi innfører derfor en pris p_i som indikerer kostnaden av bruk av fellesressurs i. Problemet (DW2) kan dermed deles opp i flere mindre problemer, såkalte *subproblemer*. Målfunksjonen i hvert av disse subproblemene vil bestå av summen av de opprinnelige kostnadene minus kostnaden for bruk av den felles ressursen. Hvert av disse subproblemene vil bestå av et av de separable restriksjonssettene (3.29) eller (3.30) samt den tilhørende nye målfunksjonen.

I vårt eksempel vil denne inndelingen gi subproblem (S1):

$$
\begin{aligned}
min \qquad & \sum_{j \in S_1} (C_j - \sum_{i=1..k} A_{ij} p_i) x_{1j} \\
\sum_{j \in S_1} B'_{ij} x_{1j} = d'_j \qquad & \forall i \in (k+1,m) \\
x_{1j} \geq 0 \qquad & \forall j
\end{aligned}
\tag{3.32}
$$

og subproblem (S2):

$$\begin{aligned} min \qquad & \sum_{j \in S_2} (C_j - \sum_{i=1..k} A_{ij} p_i) x_{2j} \\ \sum_{j \in S_2} B''_{ij} x_{2j} = d''_j \qquad & \forall i \in (m+1, n) \\ x_{2j} \geq 0 \qquad & \forall j \end{aligned} \tag{3.33}$$

Den ekstra indeksen på beslutningsvariabelen indikerer hvilken subproblem variablene tilhører.

Vi antar at løsningen i hvert av subproblemene kan angis ved et sett av hjørnepunkter, $v_{S_j k}$. S_j sier hvilket subproblem beslutningsvariabelen x_{S_j} tilhører og $k \in (1, K)$ angir hvilket hjørnepunkt. I tillegg vil hvert hjørnepunkt bli tillagt en vekt λ_{sk}. Hver av disse hjørnepunktsvektene må være ikke-negative og tilfredsstille konveksitets betingelsen:

$$\sum_{k=1}^{K} \lambda_{S_j k} = 1 \qquad \forall S_j \in \{1, S\} \tag{3.34}$$

(3.34) betyr at summen av hjørnepunktsvektene i et hvert subproblem må være lik en.

Enhver beslutningsvariabel x_j kan dermed skrives som en lineær kombinasjon av hjørnepunktene tilhørende sitt subproblem, dvs

$$x_j = \sum_{k=1}^{K} v_{S_j k} \lambda_{S_j k} \qquad \forall j \tag{3.35}$$

Masterproblemet vil her være gitt av målfunksjonen og fellesrestriksjonen i det opprinnelige problemet (DW), se (3.27), i tillegg til restriksjonene (3.34). Hvis vi substituerer den lineære kombinasjonene for hver beslutningsvariabel, (3.35), inn i målfunksjonen og fellesbeskrankningen gitt i (3.27) får vi:

$$\begin{aligned} w &= \sum_j C_j \sum_{k=1}^{K} v_{S_j k} \lambda_{S_j k} \\ \sum_j A_{ij} \sum_{k=1}^{K} v_{S_j k} \lambda_{S_j k} &= b_i \qquad \forall i \in (1, k) \end{aligned}$$

Ut i fra disse ligningene kan en definere nye målfunksjonskoeffisienter og fellesrestriksjonskoeffisienter:

$$\begin{aligned} MaC_{S_j k} &= C_j v_{S_j k} \\ MaA_{S_j k i} &= A_{ij} v_{S_j k} \end{aligned}$$

Masterproblemet vil dermed se ut som følger:

$$
\begin{aligned}
min \qquad & \sum_{S_j} \sum_{k} MaC_{S_jk} \lambda_{S_jk} \\
\sum_{S_j} \sum_{k} MaA_{S_jki} \lambda_{S_jk} \qquad & \forall i \in (1,k) \\
\sum_{k=1}^{K} \lambda_{S_jk} = 1 \qquad & \forall s \in (1,S) \\
\lambda_{S_jk} \geq 0 \qquad & \forall S_j, k
\end{aligned}
$$

Mastermodellen vil generelt ha færre beskrankninger enn den opprinnelige modellen. Antall fellesbeskrankninger vil være likt som i den opprinnelige modellen. Hvert subproblem har imidlertid blitt redusert til en beskrankning. Derimot har antall beslutningsvariable økt. Vi har nå en variabel for hvert hjørnepunkt i hver submodell. Master problemet kan løses ved å iterere mellom å løse et av subproblemene og et *redusert master problem*. Hvert subproblem som løses gir en ny kolonne for master problemet. Hver gang en løser det reduserte masterproblemet vil en få ut et sett med skyggepriser, p_i, for hver av fellesbeskrankningene. Denne prisen, p_i, blir brukt til å løse hvert av subproblemene. Hvis et eller flere av subproblemene gir en løsning som korresponderer til en kolonne som ikke allerede eksisterer i det reduserte master problemet, vil denne (disse) kolonne lagt til. Deretter blir det redusert masterproblemet løst på nytt. I et minimeringsproblem vil disse iterasjonene foregå helt til det reduserte problemet ikke har noen negativ redusert kostnad, p_i. Igjen vil det motsatte være tilfelle for et maksimeringsproblem.

3.8 Heuristikker og tabusøk

Heltallsproblemer kan være vanskelig å løse eksakt. Derfor kan en alternativt ta i bruk en eller flere *heuristikker* for å prøve å finne en løsning. Vi vil gi noen ekstra kommentarer for bruk av heuristikker i forbindelse med set partitioning problem, se kapittel 3.2. Problemet med en heuristikk er imidlertid at en ikke vet om en har oppnådd den optimale løsningen for problemet.

En *konstruktiv heuristikk* starter på null, det vil si uten verdi på noen av beslutningsvariablene. Heuristikken velger en verdi for en og en beslutningsvariabel om gangen helt til heuristikken har bygget opp en lovlig løsning. Den mest vanlige heuristikken er *grådig konstruktiv heuristikk*. I denne metoden fastsettes i hver iterasjon den beslutningsvariabelen som har det største positive bidraget til målfunksjonen gitt de beslutningsvariablene som allerede er fastsatt. Ved noen tilfeller kan det være lettere å tillate en ulovlig startløsning. Dette fordi det kan være like vanskelig å finne en lovlig løsning som å finne den optimale løsningen.

For å finne en startløsning på et set partitioning problem, kan en mulig framgangsmetode være å bruke en tilfeldig initiell løsning eller en konstruktiv heuristikk [Berg and Løkketangen].

En tilfeldig initiell løsning går ut på å finne ut hvor mange kolonner som i gjennomsnitt må til for å få en set partitioning-løsning. Dette tallet c finnes ved å dele antall rader a med hvor mange rader hver kolonne i gjennomsnitt dekker b. Denne tilfeldige heuristikken velger dermed opptil c antall kolonner tilfeldig. For hvert valg av kolonne må en sjekke at kolonnen ikke fører til overdekning. Dekker den nye kolonnen rader som allerede er dekket, slettes den fra mulighetsområdet. Denne metoden leder ikke nødvendigvis til en set partitioning løsning, og kan derfor være et dårlig utgangspunkt.

Ved bruk av konstruktiv initiell løsning for et set partitioning problem starter en ved å velge en kolonne. Denne kan for eksempel være den kolonnen som dekker flest rader per kostnad, det vil si en såkalt grådig strategi. Alternativet kan være å velge en tilfeldig kolonne. Dermed vil en få forskjellig startløsning ved å gjennom denne heuristikken flere ganger. For hver valgte kolonne slettes alle radene som denne kolonnen dekker, samt alle andre kolonner som dekker disse radene.

Ved løsing av heltallsproblemer kan en framgangsmåte være å bruke en konstruktiv heuristikk først og deretter en *forbedrende heuristikk*. En forbedrende heuristikk kan være et *lokalt søk*. Lokalt søk for et kombinatorisk optimeringsproblem kalles ofte *tilstandsrom søk* (*eng: state space search*). For en gitt målfunksjon, $f(x)$, vil et lokalt søk sjekke en løsningstilstand etter den andre. For å flytte seg fra en tilstand til en annen må en definere et sett med lovlige *bevegelser* for hver node.

I hver iterasjon vil en da sjekke det lovlige settet med naboløsninger for den noden en befinner seg i. Hvis en nabonode har en bedre løsning enn den noden en står i vil en flytte seg til denne node. Denne sekvensen stopper når ingen nabonoder har en bedre målfunksjonsverdi. Ved et slikt lokalt søk kan heuristikken stoppe i et lokalt maksimum.

En strategi for å unngå lokale optimum kan være å tillate *ikke-forbedrende bevegelser*. Da er det imidletid viktig til ethvert tidspunkt å ta vare på den beste løsningen som er oppnådd fram til nå. Det er også viktig å forhindre gjentagende løsninger slik at ikke heuristikken vil gå i en evig sirkel. I *tabu søk* håndteres dette problemet ved midlertidig å forby bevegelser tilbake til noder som nylig er besøkt. Dette gjennomføres ved å ha en *tabu liste* som inneholder ulovlige bevegelser, og i hver iterasjon må det velges en ikke-tabu mulig retning. Ved hver forflytning legges denne siste beveglsen til tabulisten slik at en ikke returnerer til denne løsningen innen et gitt antall iterasjoner. Ved tabu søk må en gjøre et valg for størrelsen for tabu-listen og implisitt dermed også et valg for hvor mange iterasjoner en bevegelsesretning skal være ulovlig.

En mulig heuristikk for lokalt søk for et set partitioning problem er å angripe problemet som et set packing problem, se kapittel 3.2. Dette lokal søket består dermed av to faser, en konstruktiv fase og en destruktiv fase [Berg and Løkketangen]. Den konstruktive fasen består i å legge til nye kolonner til løsningen, og en vil dermed gå over til set cover problem. Denne taktiske utvelgelsen vil dermed gå ut på å sikte seg inn på en set partitioning løsning. Om en ikke finner en lovlig løsningen i den konstruktive fasen beveger en seg over i den destruktive fasen. Her fjerner man da noen kolonner uten å bekymre seg for overdekning og vil dermed bevege seg tilbake til en set packing løsning. Overgangen mellom konstruktiv og destruktiv løsning itererer i håp om å finne et set partitioning løsning.

En annen metode for å tillate ikke-forbedrende bevegelse er *simulert avkjøling*. Simulert avkjøling utføres ved at i hver iterasjon velges et tilfeldig sett med lovlige bevegelser,

uavhengig at deres innflytelse på målfunksjonsverdien. Alle bevegelser som fører til noder med en bedre målfunksjonsverdi blir akseptert, mens bare noen av de ikke-forbedrende bevegelsene blir akseptert. Sannsynlighet for en disse ikke-forbedrende retningene blir godtatt avhenger av en beregnet sannsynligheten p. P er avhengig av bevegelsens negative nettobidrag til målfunksjonsverdien, δM, og av sannsynligheten q. Verdien på q vil som oftest være stor i starten av algoritmen og avta med antall iterasjoner. Dermed vil sannsynligheten p for å bli akseptert avta med økende δM og en synkende q. Denne sannsynlighet ,p, minker med antall iterasjoner og dermed synker sannsynligheten for at en ikke-forbedrende bevegelse blir akseptert. Dermed vil flere og flere ikke-forbedrende retninger blir forkastet. Som ved tabu søk kan den beste verdien, funnet med denne algoritmen, være et globalt optimum, men det finnes ingen garantier.

Kapittel 4

Litteratur om materiellplanlegging

I dette kapitlet ønsker vi å nevne noen materiellplanleggingsproblemer der vedlikehold er tatt hensyn til eller nevnt.

Problemet med å tilordne materiell til et oppsatt rutetilbud har flere navn som *train assignment*, *train rostering*, *vehicle scheduling* eller *rolling stock rostering* [Erlebach et al., 2004]. Vi vil her kalle et slikt problem et *materiellтilordningsproblem*. Som utgangspunkt for et slikt problem har man en gitt ruteplan. Problemet består dermed av å tilordne materiell til denne ruteplanen gitt en del restriksjoner. Disse restriksjonene vil bestå av krav som er nødvendig for at materiellturningen skal bli lovlig. Dette kan være krav til etterspørselen fra kundene, periodiske turneringer, sykler eller andre rammebetingelser i det aktuelle problemet. Målfunksjonen for et slikt problem består normalt i å minimere kostnadene i forbindelse med en slik tilordning. Løsningen på et materiellтilordningsproblem er som oftest en kjøreplan der hver tur i ruteplanen er tildelt materiell.

Vedlikeholdsproblemet vårt skiller seg fra et klassisk materiellтilordningsproblem ved at det allerede har bestemt materiell til ruteplanen. Se nærmere beskrivelse av vedlikeholdsproblemet i kapittel 6. Vedlikeholdsproblemet er i utgangspunktet en del av et materiellтilordningsproblem og kan dermed løses samtidig som en tilordner materiell til ruteplanen. Derfor er det interessant å se hvordan andre har tatt hensyn til denne vedlikeholdsdelen av et tilordningsproblem.

Cordeau et al. [2001b] ser på problemet ved samtidig tilordning av både lokomotiv og vogner til en ruteplan. Et slikt problem kalles *Simultaneous assignment problem*. I et slikt problem må vognene, i likhet med et vanlig tilordningsproblem, blant annet tilfredsstille krav fra kundene. Lokomotivene må tilfredsstille krav til trekkkraft fra disse vognene.

Mengden litteratur som beskriver materiellplanleggingsproblem er stor, men det har heller vært vanskelig å finne litteratur som legger vekt på vedlikehold av materiellparken. Vi har sett to hovedangrepsmåter til vedlikeholdsproblemet, tidsavhengig og distanseavhengig vedlikehold.

4.1 Tidsavhengig vedlikehold

En vedlikeholdsrestriksjon kan utrykkes ved et maksimalt antall dager som er tillatt mellom to vedlikehold.

4.1.1 Train assigment problem

Erlebach et al. [2004] beskriver en basismodell for et train assigment problem for en materielltype. Ut fra denne modellen gjør Erlebach et al. [2004] noen utvidelser, som blant annet inkluderer et vedlikeholdskrav ved at et sett av stasjoner blir designet som vedlikeholdsbaser i input. For å bli vedlikeholdt må hvert togsett periodisk innom en av disse vedlikeholdsbasene. Dette betyr at hver eneste sykel må inneholde minst en stopp på en vedlikeholdsbase.

Erlebach et al. [2004] gjør i tillegg en antagelse om at vedlikeholdet tar null tid. Dette fordi det er vanlig å ha en reserve på ca 10%, det vil si at 10% av materiellparken ikke brukes for å kjøre den oppsatte ruteplanen. Når et tog kommer inn til vedlikehold, erstattes dette derfor umiddelbart av et vedlikeholdt tog.

Løsningstiden for å løse dette basisproblemet er $O(nlogn)$. Hvis man inkluderer tomtogsforflytning vil problemet kunne løses i polynomisk tid, $O(n^2)$. Med vedlikehold vil problemet være NP-hardt.

4.1.2 Simultaneous assignment problem

Modellene til Cordeau et al. [2001b,a] gir forslag til hvordan vedlikeholdskravet kan bli håndtert. Problemet som blir løst 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 beskrankninger som binder disse sammen til et flervarenettverk, se kapittel 3.1. Cordeau et al. [2001b] utvider denne modellen til å inkludere vedlikeholdsbeskrankninger, og sier at dette vedlikeholdskravet kan utrykkes i form av et maksimalt antall tillatte dager mellom to etterfølgende stopp på en vedlikeholdsbase. Selv om kravet til vedlikehold kan bli tilfredsstilt i basismodellen, må vedlikeholdskravet i følge Cordeau et al. [2001b] normalt uttrykkes eksplisitt hvis vedlikeholdsfrekvensen er høy og antall vedlikeholdsbaser er begrenset. For å løse dette problemet erstatter Cordeau et al. [2001b] envarenettverket for hver materielltype med et flervarenettverk. Hvert av disse flervarenettverkene vil tilhøre en gitt materielltype. Varene i dette nettverket vil være bestemt av hvor lenge siden varen sist ble vedlikeholdt.

Også Cordeau et al. [2001a] nevner vedlikeholdskravet gitt som at hver sykel må inneholde et periodisk stopp på vedlikeholdsbasen. Disse stoppene må vare lenge nok for å tillate at vedlikehold og mindre reparasjoner blir utført. Når alle togene kjører om dagen, kan dette vedlikeholdet bli begrenset til å utelukkende bli gjennomført på natten. Da vil stoppet på vedlikeholdsbasen bli så langt at det alltid vil være tilstrekkelig med tid å gjennomføre vedlikeholdet. I tillegg innfører Cordeau et al. [2001a] en beskrankning som passer på at vedlikeholdskapasiteten ved vedlikeholdsbasen ikke blir overskredet.

Lingaya et al. [2002] fokuserer også på vedlikehold gitt ved et maksimalt antall dager.

4.2 Distanseavhengig vedlikehold

En vedlikeholdrestriksjon kan utrykkes ved et maksimalt antall kilometer som er tillatt mellom to vedlikehold.

4.2.1 Train assigment problem

Nõu et al. [1997] behandler et *locomotive schedule problem* for den svenske jernbanen. Et slikt problem tilsvarer egenskapene til et train assigment problem.

Nõu et al. [1997] utrykker vedlikeholdskravet ved at den akkumulerte kjørte distansen ikke må overskride den forhåndsbestemte kilometergrensen. Målet for modellen er å finne ut om de tilgjengelige lokomotivene er tilstrekkelig for å dekke gitte krav under betingelse om at vedlikeholdskravene blir oppfylt.

Nõu et al. [1997] understrekker at det er vanskelig å modellere dette vedlikeholdskravet fordi det er vanskelig å verifisere gjennomførbarheten av disse vedlikeholdsbeskrankningene på planleggingsstadiet. Nõu et al. [1997] antar at antall kjørte kilometer ved starten av perioden er uniformt fordelt. Nõu et al. [1997] skisserer at en mulig angrepsmåte for dette problemet er å relaksere vedlikeholdsrestriksjonen og deretter å planlegge vedlikeholdet i etterkant. Hvis de fleste vedlikeholdsbasene er strategisk plassert i jernbanenettet, vil de fleste syklene inneholde muligheter for vedlikehold. Problemet blir dermed enklere å løse. Videre sier Nõu et al. [1997] at fra et modelleringssynspunkt vil det være hensiktsmessig å relaksere vedlikeholdsrestriksjonene, siden vedlikehold er lettere å ta i betraktning når status er kjent for alle lokomotiv, det vil si et operasjonelt planleggingsproblem.

De har testet modellen på et testsett innenfor godstrafikk. Det består av 107 stasjoner der 7 er vedlikeholdsbaser. 4 ulike lokomotivtyper, og 2422 lokomotivforespørsler.

4.3 Litteraturen knyttet til NSB

Vi har ikke funnet noen modell eller artikkel som har hovedfokus på å minimere antall vedlikehold. Imidlertid er det en del modeller som i stedet tar hensyn til vedlikehold som en del av restriksjonene. Disse modellene inkluderer dette vedlikeholdet sammen med å finne optimal bruk av materiellparken. Kostnadene for vedlikehold er dermed inkludert som en del av driftskostnadene som en ønsker å minimere. Det som skiller vårt problem fra en del andre problem er at det i utgangspunktet er bestemt optimal materiellbruk, og vi vil da minimere antall vedlikehold når vi har materiellbruken gitt, se kapittel 5.

Som nevnt har vi sett to hovedformer for å ta hensyn til vedlikeholdet, tidsavhengig som for eksempel Cordeau et al. [2001a,b] og Lingaya et al. [2002], og kilometeravhengig som Nõu et al. [1997] og Bartlett and Charnes [1957]. NSB sitt vedlikehold er kilometeravhengig, gitt vet at hver materielltype må inn til vedlikehold før en maksimal kilometerdistanse. Det er dermed den sistnevnte tilnærmingsmåte som vil være mest relevant for oss.

Erlebach et al. [2004] gjør en antagelse om at vedlikeholdet tar null tid. Vi kan ikke gjøre denne antagelsen siden reservebeholdningen i NSB generelt er liten. Cordeau et al. [2001a] understrekker at dersom vedlikeholdsfrekvensen er høy og kapasiteten er begrenset er det viktig at vedlikeholdskravet utrykkes eksplisitt. Dette er tilfelle for NSB sin lokaltrafikk og støtter dermed under vår motivasjon for oppgaven. I likhet med Cordeau et al. [2001b] må vi passe på at vedlikeholdskapasiteten ved vedlikeholdsbasen ikke bli overskredet.

Nõu et al. [1997] passer på at antall kjørte kilometer er under den tillatte grensen. Denne modellen gir oss likevel ikke noen forklaring på hvordan vi skal håndtere nullstilling av antall kjørte kilometer etter et vedlikehold. I NSBs problem har en også en kapasitetsbeskrankning for hver vedlikeholdsbase. Modellen til Nõu et al. [1997] tar heller ikke hensyn til dette.

Nõu et al. [1997] viste at materiellplanleggingsproblemet kan forenkles ved å se bort fra vedlikeholdskravet. Dette forutsatte at vedlikeholdsbasene er strategisk plassert slik at de fleste syklene likevel kjører forbi en vedlikeholdsbase. I NSB sitt problem for lokaltogstrafikken på Østlandet har vi som nevnt to vedlikeholdsbaser, Skøyen og Sundland. De fleste togrutene passerer Skøyen, som er nærmeste stasjon til Filipstad, men det er færre som passerer Drammen som er nærmeste stasjon til Sundland. Dermed vil en slik forenkling svært sannsynlig gi en ulovlig løsning.

Kapittel 5

Modellering av vedlikeholdsproblemet

Det er to hovedtyper av modeller for løsning av *vehicle scheduling problem*. Den ene modellen basere seg på nettverk og den andre basere seg på kolonnevalg.

I dette kapitlet vil vi først avgrense problemstillingen vår ved å gjøre noen forenklinger og noen antagelser. Deretter vil vi gi en beskrivelse av vedlikeholdsproblemet til NSB og hvilke data vi trenger for å løse dette problemet. Til slutt vil vi gi matematiske modellformuleringer av problemet. Vi har lagt vekt på å prøve å løse problemet vårt ved hjelp av en nettverksformulering. I tillegg vil vi tilslutt presentere en alternativ formulering med kolonner.

5.1 Informasjon til vedlikeholdsproblemet

Før vi viser modelleringen av vedlikeholdsproblemet vil vi gjøre noen avgrensninger av problemstillingen i forhold til den komplette problemstillingen til NSB. Vi vil dermed gi en beskrivelse av problemet som vi har valgt å løse og hvilke data som er nødvendig for å løse problemet vårt.

5.1.1 Avgrensning av problemstillingen

Vi har gjort noen antagelser og forenklinger for å avgrense problemstillingen vår.

Planleggingshorisont og geografisk avgrensning

Vi har valgt å avgrense problemet til å se på langsiktig planlegging og til å se på lokaltogstrafikken på Østlandet. Grunnen til at vi valgte langsiktig planlegging er fordi dette var ønskelig fra NSB sin side. I utgangspunktet er prinsippene for driftspausebasert vedlikeholdsplanlegging for langdistansetrafikk og nærtrafikk like, og modellen vår vil derfor i praksis kunne brukes for begge problemstillingene. Vi har likevel tatt utgangspunkt i nærtrafikken, siden det er her det er størst mulighet for forbedring fordi det er større valgmuligheter for hvordan vedlikeholdet skal plasseres. I tillegg ønsket vi å se oppgaven vår i sammenheng med oppgaven til Aschehoug and Fodstad [2002] som har gjort tilsvarende avgrensninger.

Driftspausebasert vedlikehold

Vi har sett bort i fra tungt vedlikehold. Ved tungt vedlikehold tas togene ut av drift og erstattes med andre togsett. Derfor er dette vedlikeholdet både tids- og ressurskrevende. I tillegg forutsetter dette at vi har tilstrekkelig med tog å sette inn for de togene som blir tatt ut. Vår oppgave er begrenset til å se på vedlikehold som blir gjennomført i pauser i materiellturneringen. Dette vedlikeholdet kalles driftspausebasert vedlikehold.

Vedlikehold

Ved hver vedlikeholdsbase er det skift der det er mulighet for et eller flere vedlikehold. Vi har antatt at hvert vedlikehold må gjennomføres innen et skift og at det ikke er mulig å dele opp et vedlikehold i flere deler. Det betyr at en vil fullføre et vedlikehold før en starter på et nytt. Som en forenkling har vi derfor delt opp hvert skift i tidsintervaller som er lik lengden av et vedlikehold. I likhet med NSB har vi antatt at alle vedlikehold tar fire timer. Det betyr at hvis et skift varer fra 08.00 til 16.00, har vi delt dette skiftet opp i to muligheter for vedlikehold. Hver av disse mulighetene vil dermed betraktes som uavhengige vedlikehold. I praksis har NSB ikke denne strenge inndelingen i mulige vedlikehold. Dermed vil forenklingen vår redusere fleksibiliteten av mulige vedlikehold, men vil ikke føre til et vi får en ulovlig løsning i forhold til virkeligheten.

Hver vedlikeholdsbase kan vedlikeholde en eller flere materielltyper. Vi har antatt at hvert skift er uavhengig av materielltype og at det tar like lang tid å vedlikeholde alle materielltypene. Dette er en antagelse som NSB også selv gjør.

Prisen på vedlikehold er kilometeravhengig, og dermed i utgangspunktet uavhengig av om vedlikeholdet gjennomføres på natt eller dagtid. Det er likevel ønskelig at det meste av vedlikeholdet skjer på dagtid. Vi har i vår modell sett bort fra dette. Dette ønsket kan enkelt tas hensyn til ved å sette kostnaden på vedlikehold på natten høyere enn vedlikehold på dagen.

Ved vedlikeholdsbasene er det tilstrekkelig med spor slik at vi kan se bort ifra rekkefølgen for togene inn og ut. Overnattingskapasiteten og hensettingskapasiteten på dagstid er også ubegrenset på vedlikeholdsbasene, så vi trenger ikke ta hensyn til dette.

Kostnader

For hver uke har vi satt kostnadene for vedlikehold og tomtogskjøring slik at summen av kostnadene for all tomtogskjøring vil være mindre enn kostnaden ved et vedlikehold. Dette fordi hovedmålet vårt er å minimere antall vedlikehold og deretter tomtogskjøring. Disse kostnadene vil derfor ikke gjenspeile de reelle kostnadene, men beregnes slik at vi oppnår ønsket målsetning for modellen.

Toglengde og Etterspørsel

Som en del av inndataene til problemet har vi gitt hvilken materielltype som dekker hvert dagsløp. Disse materielltypene har en bestemt toglengde og setekapasitet. Dermed vil disse dagsløpene tilfredsstille restriksjonene til toglengde og til setekapasitet. I vår problemstilling vil vi kun koble sammen dagsløp eller deler av dagsløp. Vi tillater imid-

lertid kun koblinger av dagsløp som tilhører samme materielltype. Dermed trenger vi ikke ta hensyn til disse to restriksjonene i vår problemstilling.

Hensettingskapasitet og Sportilgang

Dagsløpene i inndataene har vi antatt tar hensyn til hensettingskapasiteten på dagtid. Siden vi tillater oss å dele disse dagsløpene i mindre dagsløp kan dette føre til endringer for ved hvilke stasjoner togene blir stående på dagstid. Siden vi ikke har noen oversikt over hvor de ulike togene befinner seg på dagtid, vil ikke disse dataene gi oss mulighet til å sjekke om dette kravet fortsatt er tilfredsstilt. Vi vil derfor se på bort fra hensettingskapasiteten.

Dagsløpene i inndataene inneholder alle togturene gitt i togtabellen. For alle disse turene er det naturlig nok sjekket om det er ledig sporkapasitet (slots). Som nevnt vil vi dele opp en del av disse dagsløpene i mindre deler og dette kan føre til at det vil bli satt opp nye tomtogskjøringer. For at disse tomtogskjøringene skal være lovlige krever det ledig slots. Siden vi ikke har noen oversikt over når det er ledige slots, har vi derfor sett bort fra dette kravet. Dette vil i praksis si at vi kan få en ulovlig løsning. Etter at en har løst problemet vårt kan en imidlertid sjekke om det er ledig slots for de nye tomtogskjøringene. Finnes ikke dette vil en forhåpentlig kunne rette på dette ved å gjøre små manuelle justeringer. Hvis en på forhånd vet at det er noen strekninger som er sterkt belastet ved gitte tidspunkt kan en eventuelt legge til restriksjoner som forhindrer tomtogskjøring for disse strekningene.

Begge disse to forenklingene gjør av vi kan få løsninger på problemet som ikke er lovlige i virkeligheten. Vi antar at disse ulovlighetene likevel kan rettes på ved små manuelle korrigeringer og derfor vil modellen vår likevel ha verdi.

Overnattingskapasitet

Vi har i tillegg antatt at overnattingskapasiteten ved stasjonene kun er avhengig av antall tog og dermed ikke avhengig av materielltype. Denne antagelsen fører også til at vi kan få en ulovlig løsning. Denne forenklingen kan tas hensyn til ved å legge til overnattingsrestriksjoner som tar hensyn til materielltype. Vi har ikke lagt vekt på dette i modellen vår.

Som nevnt vil noen av disse forenklingene i praksis kunne føre til at vi kan få ulovlige løsninger. Noen av disse restriksjonene kan sjekkes manuelt etter at problemet er løst og forhåpentlig korrigeres for ved små endringer. Vi kunne eventuelt lagt til flere restriksjoner til modellen vår. Siden hovedmålet for oss har vært å lage en modell som tar hensyn til driftspausebasert vedlikehold har vi derfor valgt å se bort fra noen av disse kravene som bare vil komplisere problemet. For en videre utvikling av modellen kan en heller legge til flere restriksjoner. Vi mener likevel at modellen vår vil være av verdi fordi den gir en tilnærming til hvordan en kan løse problemet med driftspausebasert vedlikehold.

5.1.2 Mål

Hovedfokus for vår problemstilling er å knytte sammen dagsløpene til gyldige turneringer slik at en får en optimal tilpasning av det driftspausebaserte vedlikeholdet. I vårt

problem vil en optimal tilpasning av vedlikehold bety å minimere antall vedlikehold. I tillegg ønsker vi at antall kilometer tomtogskjøring skal bli så lavt som mulig.

5.1.3 Restriksjoner

Vi har tatt hensyn til en rekke restriksjoner i vår modellering av vedlikeholdsproblemet.

Flyt

Antall tog som kommer inn på en stasjon eller vedlikeholdsbase må være likt antallet som kjører ut igjen.

Sykel

Turneringen må være syklisk. Dette betyr at enhver kobling mellom to dagsløp vil gjenta seg etter en viss tid. Antall dager mellom hver gang en kobling gjentar seg vil være lengden av denne turneringen.

Ukesløp

NSB ønsker like grunnuker. Dette betyr at koblingene mellom to dagsløp skal være lik hver uke, det vil si hvis to dagsløp kobles sammen en uke skal de også kobles sammen neste uke. På denne måten får NSB en lik grunnuke som kan kjøres for hele planleggingshorisonten, som i dette tilfelle er et halvår.

Tid

For at to dagsløp skal kunne kobles sammen må det ene dagsløpet avsluttes før det neste starter. Hvis disse to dagsløpene slutter og starter på forskjellig stasjon må det i tillegg være tilstrekkelig med tid til å forflytte seg mellom disse stasjonene. Tilsvarende gjelder for kobling mellom vedlikehold og dagsløp.

Kilometergrense

For hvert tog må ikke antall kilometer hvert tog kjører mellom to vedlikehold overskride den maksimale kilometergrensen. Antall kilometer et tog kjører inkluderer både kjøring av dagsløp og eventuell tomtogskjøring. Denne tomtogskjøringen innebærer både kjøring mellom stasjoner der sluttstasjonen for et dagsløp er forskjelling fra startstasjonen for etterfølgende dagsløp, og kjøring mellom et dagsløp (endestasjonen) og en vedlikeholdsbase. Dette er en hard restriksjon som ikke kan overskrides under noen omstendighet.

Vedlikeholdskapasitet

Ved hver vedlikeholdsbase er det en del skift der en kan vedlikeholde tog. Hvert av disse skiftene har en kapasitet på hvor mange tog som maksimalt kan vedlikeholdes. For å sende et tog til vedlikehold må materiellplanleggeren sjekke om det er ledig kapasitet ved vedlikeholdsbasen.

Materiellplanleggeren har et bestemt antall tog av hver materiellplanleggeren tilgjengelig til disposisjon. Antall tog som trengs for å gjennomføre en sykel vil være bestemt av lengden på sykelen. Hvis en sykel består av n uker trenger man n tog for å gjennomføre denne sykelen. Totalt antall tog som brukes vil derfor være summen av antall tog for alle sykelene. Summen av alle togene per materiellplanleggeren som brukes må ikke overskride den grensen.

Overnattingskapasitet

Hver stasjon har en grense for hvor mange tog som kan overnatte.

Kompatible dagsløp

Hvert dagsløp tilhører en bestemt materiellplanleggeren. Derfor må vi passe på vi bare kobler sammen dagsløp som tilhører samme type.

5.1.4 Inndata til vedlikholdsproblemet

Dette er et første forsøk i å inkludere vedlikehold i et beslutningsstøttesystem for planlegging i NSB, og vi har derfor valgt å forhåndsbestemme hvilke materiellplanleggeren som hører til de ulike dagsløpene. Dette har vi gjort for å få fokus på vedlikeholdsproblematikken i tillegg til at det for et par år siden ble gjennomført et prosjekt for NSB der en modell for å minimere materiellbruk gitt et rutetilbud ble laget.

Aschehoug and Fodstad [2002] har formulert et planleggingsproblem for NSB med flere materiellplanleggeren. De ser på lokaltogstrafikken i Oslo, der de kjører togsett av ulike typer og subtyper. Modellen til Aschehoug and Fodstad [2002] bestemmer optimal materiellbruk. Ved utarbeidelsen av denne modellen er det gjort en del forenklinger i forhold til det komplette problemet i NSB. Derfor har de sett bort fra en del av restriksjonene til det komplette problemet. Modellen behandler ikke vedlikehold, sportilgang og snutid som er avhengig av toglengden.

Modellen tar en gitt ruteplan som inndata. Følgende data blir gitt ut av modellen indirekte eller direkte:

- hvilken materiellplanleggeren og hvor mange sett av materiellplanleggeren som dekker hver tur
- hvor og når materiellet skjøtes og deles
- hvor materiellet stalles nattestid/overnatter
- antallet av hver materiell type som befinner seg på hver stasjon til enhver tid
- forbruket av hver materiellplanleggeren

Utgangspunktet for oppgaven vår var at vi ønsket vi å bruke denne modellen til Aschehoug and Fodstad [2002] til å generere lovlig dagsløp. Siden denne modellen ikke blir brukt i praksis ennå, valgte vi i stedet å ta utgangspunkt i de planlagte turneringene som NSB skal bruke fra januar 2005. Dette valget av data har ingen betydning for

modellen vår, men vi ønsket å bruke reelle data siden vi da kan se hvordan vår løsningen stemmer overens med virkeligheten.

Imidlertid har vi gjort noen endringer i disse dataene for å øke fleksibilteten i modellen vår. I alle dagsløp som inneholder tomtogskjøring har vi fjernet denne kjøringen og delt dagsløpet opp i mindre selvstendige dagsløp hvis det er plass til et vedlikehold inne i et dagsløp. Disse nye dagsløpene vil dermed få henholdsvis sluttid og startid, før og etter denne tomtogskjøringa. I tillegg har vi fjernet tomtogskjøringen i starten og slutten av alle dagsløpene. Vi vet dermed at disse dagsløpene vil kunne gi oss gyldige turneringer.

Følgende data tar vi inn i modellen vår:

Dagsløp

Et dagsløp vil være gitt av en startstasjon og en endestasjon, et starttidspunkt og et sluttidspunkt. For hvert dagsløp er det også gitt hvilken togtype (= materielltype) som skal gjennomføre det. Hvert dagsløp vil i tillegg tilhøre en gitt ukedag. Noen dagsløp kjører over midnatt og vil dermed starte og slutt på forskjellig dag. Vi har derfor valgt å sette denne ukedagen til den dagen som dagsløpet starter. Hvert dagsløp har en gitt lengde gitt i antall kilometer.

Vi vil også tillate såkalte tomme dagsløp der kilometerlengden er lik null og som har et ubestemt start og sluttidspunkt. Dette betyr at vi har et dagsløp der toget står stille hele dagen. Vi har innført disse tomme dagsløpsnodene fordi vi i modellen krever at en turnering skal inneholde minst et dagsløp hver dag. I helgene er det mindre rutetilbud enn det er på hverdagene og dermed trengs det mindre materiell. Da trenger vi tomme dagsløp for at vi skal få en lovlig løsning. I tillegg kan vi legge til tomme dagsløp på hverdagene hvis det er ønskelig. Dette gir oss større fleksibilitet i sammenkoblingene.
Vi trenger dermed følgende informasjon om hvert dagsløp:

- Starttidspunkt
- Sluttidspunkt
- Startstasjon
- Endestasjon
- Ukedag
- Togtype
- Diagramnummer
- Lengden i antall kilometer

Togtype

Hver togtype har en gitt kilometergrense som sier hvor mange kilometer toget kan kjøre mellom hvert vedlikehold. I tillegg kan hver togtype bare vedlikeholdes ved bestemte vedlikeholdsbaser.
Vi trenger dermed følgende informasjon om hver togtype:

- Kilometergrense
- Vedlikeholdsbaser

Vedlikeholdsbaser

På hver vedlikeholdsbase kan det kun vedlikeholdes bestemte togtyper. På hver vedlikeholdsbase vil det kun være tillatt å utføre vedlikehold innen visse skift. Vi trenger dermed en oversikt over hvor mange tog som kan vedlikeholdes innen hvert skift, og når hvert skift starter og slutter.
Vi trenger dermed følgende informasjon om hver vedlikeholdsbase:

- Kapasitet for hvert skift
- Start og sluttidspunkt for hvert skift
- Togtyper

Stasjon

Ved hver stasjon er det begrenset hvor mange tog som kan hensettes til overnatting. Denne overnattingskapasiteten kan variere fra ukedag til ukedag. Ved noen stasjoner ønsker man for eksempel ikke å sette togene til overnatting i helgene. For å kunne koble sammen dagsløp og vedlikehold må vi holde rede på hvor mange kilometer et tog har kjørt. Vi vet lengden av dagsløpene, men vi må også vite lengden av tomtogskjøringen. Derfor må vi ha en oversikt over hvor langt det er mellom alle stasjonene og mellom stasjonene og vedlikeholdsbasene.
Vi trenger dermed følgende informasjon om hver stasjon:

- Overnattingskapasitet for hver ukedag.
- Avstandene til de andre stasjonene og vedlikeholdsbasene.

5.2 Nettverksformulering

Her er problemet modellert som et nettverk.

Noder og Kanter

Nodene inneholder aktiviteter og kantene er mulige koblinger mellom disse aktivitetene. Vi har to typer noder, dagsløpnoder og vedlikeholdsnoder. Dagløpnodene vil bestå av kjøring av de ruteplanlagte turene eller venting. Hver av turene i den oppsatte ruteplanen vil dermed bli inkludert i en og bare en dagløpsnode. Hver vedlikeholdsnode inneholder en mulighet for et vedlikehold.

Kantene i nettverket vil bestå av koblinger mellom disse nodene. Vi har vi valgt å tillate sammenkoblinger mellom noder som har forskjellig ende og start stasjon. Vi har som en del av vår inndata en tabell med avstandene mellom alle stasjonene. Ut fra denne tabellen beregnes antall kilometer tomtogskjøring som er nødvendig for å koble sammen

noder. Hvis det er stasjoner der vi ikke ønsker å tillate tomtogskjøring kan vi sette denne kilometerverdien veldig høy. Dette kan for eksempel være fordi denne tomtogskjøringen tilhører en toglinje som allerede er sterkt belastet. Disse kantene vil dermed ha lengden lik tomtogskjøringen som er nødvendig for å koble de tilordnede nodene sammen.

I dette kapitlet vil vi først presentere en basismodell for en togtype, *Entypeproblemet*. Denne modellen gjelder for en grunnuke og for en togtype. Dette problemet er en form for envareproblem, se kapittel (3.1.4). Vi har deretter utvidet denne modellen til en modell som tar hensyn til flere togtyper, *FlertyperProblemet*. Denne modellen er en form for *flervareflytproblem* siden hensettingskapasiteten ved stasjonene og vedlikeholdskapasitetene er avhengig av flere togtyper. Denne har vi til slutt utvidet til en modell som tar hensyn til flere togtyper, *FlertyperProblemet*. Denne siste modellen tilfredsstiller kravene til vår problemformulering av planleggingsproblemet til NSB, se kapittel (5.1.2). Navnene på modellene er valgt ut i fra hvilken ny funksjonalitet modellen innfører.

5.2.1 Notasjon

Nedenfor følger den notasjonen vi har brukt i modellene våre. Vi har gjengitt all notasjonen samlet og noe av notasjonen blir gjentatt flere ganger, bare med ulike indekser.

Indekser:

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

Sett:

N	Noder i nettverket
S	Stasjoner i nettverket
K	Materielltyper
N^k	Noder der materielltype k kan benyttes.
D	Dagsløpsnoder i nettverket, $D \in N$.
D^k	Dagsløpsnoder som kan kjøres av materielltype k, $D^k \in N^k$
V	Vedlikeholdsnoder i nettverket, $V \in N$.
V^k	Vedlikeholdsnoder der materielltype k kan vedlikeholdes, $V^k \in N^k$
O_j	Lovlige etterfølgernoder (utnoder) for node j, $O_j \in N$.
I_j	Lovlige forgjengernoder (innoder) for node j, $I_j \in N$.
P	Dager i planleggingsperioden, $P \in \{1, PTOTAL\}$.
G	Dager i en grunnuke, $T \in \{1, TTOTAL\}$

Konstanter:

$KMTT_{ij}$	Avstanden i kilometer mellom endestasjon i node i og startstasjon i node j. For dagsløpnoder der endestasjon for i er lik startstasjon for j vil $KMTT_{ij}$ være null. Ellers vil denne lengden angi tomtogskjøring.
$TIDTT_{ij}$	Avstanden i tid mellom endestasjon i node i og startstasjon i node j. For dagsløpnoder der endestasjon for i er lik startstasjon for j vil $TIDTT_{ij}$ være null. Ellers vil denne lengden angi tiden det tar å kjøre denne tomtogskjøringen.
$KMGRENSE$	Maksimalt antall tillatte kjørte kilometer mellom to vedlikehold.
$KMGRENSE_k$	Maksimalt antall tillatte kjørte kilometer mellom to vedlikehold for materielltype k.
$LENGDE_j$	Tabell med lengden av dagsløpsnode j 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.
T_j	Ukedag for dagsløp j.
$HKAP_s$	Maksimalt antall sett det er lov til å hensette ved stasjon s.
$MATKAP$	Tilgjengelig materiell når vi bare har en togtype.
$MATKAP_k$	Tilgjengelig materiell av type k.
$PTOTAL$	Antall dager i planperiode.
$TTOTAL$	Antall dager i en grunnuke.
$UTOTAL$	Antall uker i en planperiode.

C^V Kostnad ved et vedlikehold.

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

Beslutningsvariable:

Flytvariabelen:

$$x_{ij} = \begin{cases} 1 & \text{hvis node } i \text{ kobles til node } j \\ 0 & \text{ellers} \end{cases}$$

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

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

Kilometervariabelen:

z_j= avstanden siden forrige vedlikehold gitt at dagsløp j er gjennomført

z_{ju}= avstanden siden forrige vedlikehold gitt at dagsløp j er gjennomført i uke u

Hensettingsvariabelen:

y_{st}= antall sett som blir hensatt på stasjon s på dag t

y_{st}^k = antall sett som blir hensatt på stasjon s på dag t av type k

y_{stu}^k = antall sett som blir hensatt på stasjon s på dag t av type k i uke u

Materiellvariabelen:

w = antall tog som trenges ved en materielltype

w^k = antall tog som trenges av materielltype k

5.2.2 Generering av subsett

Inndataene til vårt problem gjelder for en uke. Hvis vi ønsker å bruke modellen vår på en planperiode som består av u grunnuker løser vi dette ved å kopiere opp nodene for alle ukene. Dermed vil hver uke inneholde de samme nodene, bortsett fra at de er indeksert med u som sier i hvilken uke noden befinner seg i. I dette nettverket er det flere krav som begrenser hvilke noder som kan kobles sammen. Derfor lager vi for hver node j subsettene I_j og O_j. Subsettet I_j er lovlige forgjengere og subsettet O_j er lovlige etterfølgere til node j. Vi vil også kalles disse settene henholdsvis *innoder* og *utnoder*.

For at node i skal være en lovlig forgjenger til node j må vi ha følgende restriksjoner:

1. Komplatible typer

Det er kun tillatt å koble sammen node i og node j dersom de tilhører samme togtype k. For hver vedlikeholdsnode har vi i tillegg satt et krav på at for alle vedlikeholdsnoder j så må node i være en dagsløpsnode for å få være en lovlig innode. Dette kravet er strengt tatt ikke nødvendig. Siden vi ønsker å minimere antall vedlikehold vil dette kravet høyst sannsynlig bli oppfylt uansett. Vi innfører det likevel for å gjøre settet I_j så lite som mulig.

2. Dager

For at node i skal være en lovlig innode til j så må enten node i tilhøre dagen før node j, eller node i og j må tilhøre samme dag. Hvis planperioden består av en uke betyr dette at hvis node j tilhører dag t må node i tilhøre dag t eller $(t-1)$. For noder på første dag i uken vil imidlertid lovlige innoder blant annet inneholde noder fra siste dag i uken. Når planperioden består av flere grunnuker, må vi på tilsvarende måte sørge for at lovlige innoder for første dag i planperioden blant annet består av noder på siste dag i planperioden. I tillegg må vi sørge for lovlige koblinger mellom hver grunnuke. Dette betyr at de lovlige innodene til node j, som tilhører første dag i en grunnuke, i uke u blant annet inneholder noder i i siste dag i grunnuke $u-1$.

Med dette kravet sier vi at vi ikke ønsker å koble to noder som har noen mellomliggende dager. Dette ville i praksis betydd at toget ville stått stille i disse dagene. Vi tillater imidlertid dette, men vi løser det i stedet for ved å tillatte tomme dagsløp.

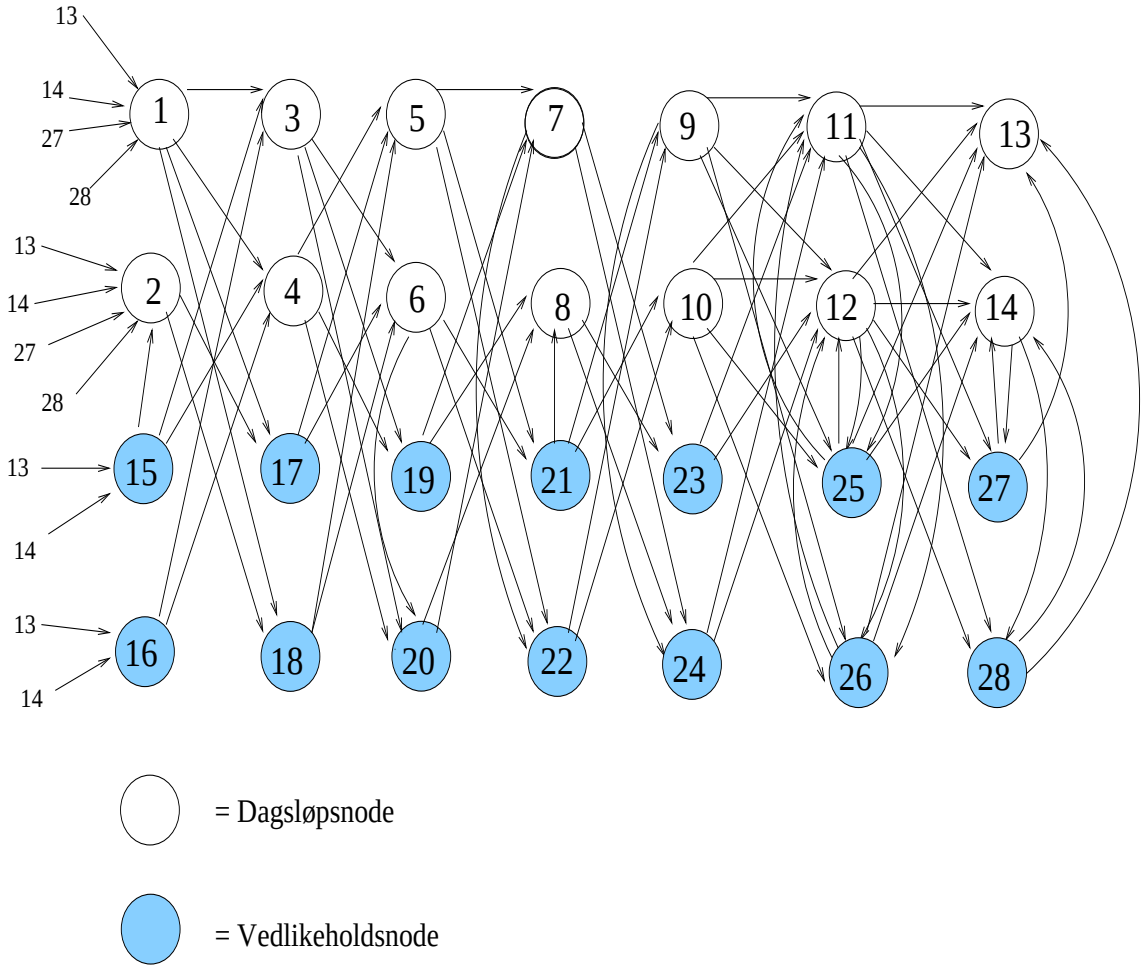
3. Tid

$$SLUTTID_i + TIDTT_{ij} \leq STARTTID_j \tag{5.1}$$

Dette kravet sørger for at tidsrestriksjonen er oppfylt. For at node i skal være en lovlig innode til node j må tiden mellom avslutningstidspunktet for node i, $SLUTTID_i$ og starttidspunktet for node j, $STARTTID_j$, være større en tiden det tar å kjøre mellom i og j, $TIDTT_{ij}$.

Tilsvarende beskrankninger må være oppfylt for at node i er en lovlig utnode for node j. Kort oppsummert betyr dette at dersom node i skal være en lovlig utnode til node j så må nodene tilhøre samme togtype k, og lovlige utnoder for en vedlikeholdsnode

må være en dagsløpsnode. En lovlig utnode i må tilhøre samme dag eller dagen etter node j. For noder på siste dag i planperioden må utnodene blant annet inneholde noder på første dag i planperioden. I tillegg må tiden mellom avslutningstidspunktet for node j, $SLUTTID_j$ og starttidspunktet for node i, $STARTTID_i$, være større en tiden det tar å kjøre mellom j og i, $TIDTT_{ji}$.



Figur 5.1: Nettverket for inndatasettet typer1

Nettverket for dette problemet vil dermed bestå av rettede kanter fra alle innoder, I_j, til node j, og fra node j til alle lovlige utnoder, O_j, se figur 6.1. Flytvariabelen, x^k_{iju}, vil kun være definert over disse kantene. Dette gjøres for at antallet beslutningsvariable skal være så lavt som mulig.

5.3 EntypeProblemet

Denne modellen er forenklet i forhold til vår formulering av vedlikeholdsproblemet blant annet fordi den kun tar hensyn til en togtype.

5.3.1 Målfunksjonen

Vi har følgende målfunksjon i EntypeProblemet:

$$min \sum_{i \in I_j} (\sum_{j \in V} C^V x_{ij} + \sum_{j \in N} C^{TT} KMTT_{ij} x_{ij}) \tag{5.2}$$

(5.2) er målfunksjonen som minimerer kostnaden ved vedlikehold, C^V, og kostnaden ved tomtogskjøring, C^{TT}, for èn uke. Kostnaden for tomtogskjøring er definert slik at den totale kostnaden for tomtogskjøring vil være mindre en kostnad ved et vedlikehold. Dette har vi gjort fordi hovedfokuset vårt er å minimere antall vedlikehold. Antallet vedlikehold blir beregnet ved å summere over alle innodene, I_j, som går inn til alle vedlikeholdsnodene, $j \in V$, der flyten over kant ij er lik èn. Kostnaden for tomtogskjøring blir beregnet ved å summere over innodene, I_j, som går inn til alle dagsløpsnoder og alle vedlikeholdsnoder, $j \in N$, der flyten over kant ij er lik èn. For vårt problem er det ikke nødvendig å ta med kostnadene for å kjøre dagsløpene. Denne kostnaden er konstant og vil derfor ikke påvirke optimeringen.

5.3.2 Beskrankninger

Restriksjonene fra problemformuleringen vår er blitt tatt hensyn til på følgende måte:

Flytrestriksjonen

$$\sum_{i \in I_j} x_{ij} - \sum_{i \in O_j} x_{ji} = 0 \qquad \forall j \in N, \tag{5.3}$$

$$\sum_{i \in I_j} x_{ij} = 1 \qquad \forall j \in D, \tag{5.4}$$

$$\sum_{i \in O_j} x_{ji} = 1 \qquad \forall j \in D \tag{5.5}$$

$$\sum_{i \in I_j} x_{ij} \leq 1 \qquad \forall j \in V \tag{5.6}$$

(5.3) er *flytkonserveringsrestriksjon*. Denne restriksjonen sier at for alle noder, j, må flyten inn til noden være lik flyten ut av noden. Restriksjon (5.4) sørger for at flyten inn i hver dagsløpnode, j, er lik en. Tilsvarende sørger (5.5) for at flyten ut av hver dagsløpsnode er lik en. Disse to restriksjonene sørger derfor for at alle dagsløpsnoder, j, blir dekket en gang. (5.6) sørger for at maksimal tillatt flyt inn til en vedlikeholdsnode er lik en. Dette betyr at hver vedlikeholdsnode høyst kan besøkes en gang.

På grunn av flytkonserveringsrestriksjonen (5.3) og restriksjon (5.4) er restriksjon (5.5) automatisk oppfylt og dermed redundant. Alternativt kunne man ha sagt at (5.4) er redundant. Denne restriksjonen kan likevel være smart å ta med hvis man vil løse problemet ved å Lagrange relaksere restriksjon (5.3) [Huisman et al., 2003]. Da vil det være nødvendig å ha begge. Vi vil i dette prosjektet ikke løse problemet ved hjelp av Lagrange relaksering og vil derfor fjerne restriksjon (5.5) i resten av denne rapporten.

Sykliske Turneringer, Tid og Like ukesløp

Ut fra subsettene I_j og O_j vet vi at lovlige utnoder for en node j på siste dag i grunnuken blant annet vil inkludere noder på første dag i grunnuken. Tilsvarende vil lovlige innoder for en node j på første dag i grunnuken blant annet inneholde alle nodene på siste dag i grunnuken, se kapittel (5.2.2). Flytrestriksjonen (5.3) sier at hver dagsløpsnode må kobles en gang. Dermed vil noder på en mandag kunne bli koblet sammen med noder på en søndag. Vi ser dermed at subsettene I_j og O_j sammen med Flytrestriksjonen sørger for at kravet til *Sykliske turneringer* blir oppfylt.

Ved generering av disse subsettene sørget vi for at restriksjonen *Tid* ble oppfylt. Siden denne modellen kun gjelder for en grunnuke vil kravet om *Like ukesløp* automatisk være oppfylt. Derfor blir ikke dette kravet uttrykt eksplisitt i denne modellen.

Vedlikehold

For å oppfylle vedlikeholdsbeskrankningen formulerte vi først noen kvadratiske restriksjoner. Nõu et al. [1997] bruker en restriksjon som likner på våre vedlikeholdsrestriksjoner, bortsett fra at deres restriksjon ikke har inkludert tomtogskjøring. I vårt problem vil denne restriksjonen se forskjellig ut avhengig av hvilke noder man forlater og entrer.

Fordi en helst vil unngå slike ulineære formuleringer har vi omformulert dem til å bli lineære. Vi viser først hvordan de ulineære restriksjonene ser ut, før vi viser hvordan de blir omformulert og forklarer hva de gjør.

$$x_{ij}(z_i + KMTT_{ij} + LENGDE_j - z_j) \leq 0 \qquad \forall i \in D \cap I_j, j \in D \tag{5.7}$$

$$x_{ij}(z_i + KMTT_{ij} - KMGRENSE) \leq 0 \qquad \forall i \in D \cap I_j, j \in V \tag{5.8}$$

$$x_{ij}(KMTT_{ij} + LENGDE_j - z_j) \leq 0 \qquad \forall i \in V \cap I_j, j \in D \tag{5.9}$$

$$0 \leq z_j \leq KMGRENSE \qquad \forall j \in D \tag{5.10}$$

$$x_{ij} \in \{0, 1\} \qquad \forall j \in N, i \in I_j \tag{5.11}$$

Den første linjen i omformuleringen bruker vi til å illustrere hva hver restriksjon betyr, men i den andre linjen har vi ordnet ligningen slik at beslutningsvariablene står på venstre side av likhetstegninget og konstantene står på høyre side av likhetstegnet. Restriksjon (5.7), (5.8) og (5.9) blir henholdsvis omformulert som følger:

$$\begin{aligned} &z_i + KMTT_{ij} + LENGDE_j - z_j \leq M1_{ij}(1 - x_{ij}) \\ &z_i - z_j + M1_{ij}x_{ij} \leq M1_{ij} - KMTT_{ij} - LENGDE_j \\ &\qquad\qquad \forall i \in D \cap I_j, j \in D \end{aligned} \tag{5.12}$$

$$z_i + KMTT_{ij} + KMGRENSE \leq M2_{ij}(1 - x_{ij})$$
$$z_i + M2_{ij}x_{ij} \leq M2_{ij} - KMTT_{ij} - KMGRENSE$$
$$\forall i \in D \cap I_j, j \in V \tag{5.13}$$

$$KMTT_{ij} + LENGDE_j - z_j \leq M3_{ij}(1 - x_{ij})$$
$$M3_{ij}x_{ij} - z_i \leq M3_{ij} - KMTT_{ij} - LENGDE_j$$
$$\forall i \in V \cap I_j, j \in D \tag{5.14}$$

Teoretisk sett kan vi velge en hvilken som helst verdi M for hver restriksjon, gitt at denne verdien for M gjør restriksjonen redundant for x_{ij} lik null. Grunnen til at vi velger M så liten som mulig er for å gjøre restriksjonene så strenge som mulig ved x_{ij} lik null. Dette gjør at løsningsrommet blir så lite som mulig og kan dermed bidra til å redusere løsningstiden for problemet. Derfor beregner vi verdier som er avhengig av node i og node j for M'ene.

Beregning av $M1_{ij}$, $M2_{ij}$ og $M3_{ij}$ for hhv (5.12), (5.13) og (5.14) :

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

$$M2_{ij} = max\{z_i + KMTT_{ij} - KMGRENSE\} = KMTT_{ij}$$

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

Restriksjonene (5.12) til (5.14) er alle redundante hvis x_{ij} er lik null og vil da ikke ha noen innvirkning på løsningen. Ved x_{ij} lik en vil disse restriksjonene sørge for at vedlikeholdskravet bli tilfredsstilt.

(5.12) teller antall kjørte kilometer ved tilordningen mellom to dagsløpsnoder i og j. Antall kjørte kilometer for dagløpsnoden j, z_j, blir satt lik antall kjørte kilometer for forgjenger noden i, z_i, pluss eventuell tomtogskjøring mellom nodene, $KMTT_{ij}$ og dagsløpslengden for node j, $LENGDE_j$.

(5.13) teller antall kjørte kilometer ved tilordning mellom en dagslopsnode i og en vedlikeholdsnode j. Denne restriksjonen sørger for at antall kilometer for dagsløpsnoden i, z_i, pluss eventuell tomtogskjøring til vedlikeholdsbasen j, $KMTT_{ij}$, ikke overskrider den maksimale tillatte kilometergrensen, $KMGRENSE$.

(5.14) teller antall kjørte kilometer mellom en vedlikeholdsnode i og en dagsløpsnode j. Etter at et tog har vært innom vedlikehold starter naturlig nok tellingen av kilometer på nytt. Denne restriksjonen sørger dermed for at kilometertelleren, z_j, blir nullstilt når

toget kjører fra vedlikeholdsbasen. Antall kilometer for en dagslopsnode j som etterfølger en vedlikeholdsnode i vil derfor være gitt av eventuell tomtogskjøring, $KMTT_{ij}$, til noden og lengden av dagsløpnoden j, $LENGDE_j$.

Grunnen til at vi kan formulere restriksjonene (5.7)-(5.9) og dermed også (5.12)-(5.14) som ulikheter er fordi vi har et minimeringsproblem og kan tillate slakk. Det vi må sørge for er at kilometertelleren minimum teller antall kjørte kilometer. Dette kan illustreres ved at for eksempel antall kjørte kilometer i en dagsløpsnode j er A. Hvis etterfølgende node for j er en vedlikeholdsnode og avstanden til denne noden er B kilometer, kan kilometertelleren for node j ligge i intervallet $\{A, KMGRENSE - B\}$. Dette betyr at hvis man vil telle opp antall kilometer kjørt, må man beregne dette uten å bruke kilometertellervariabelen.

(5.10) sier at kilometertelleren, z_j, for node j må ligge mellom null og kilometergrensen, $KMGRENSE$. For alle vedlikeholdsnoder er antall kjørte kilometer lik null. Derfor er kilometervariabelen, z_j kun definert for alle dagsløpsnoder. (5.11) gir binærkravet til flytvariabelen, x_{ij}.

Restriksjonene (5.12)-(5.14) gjør at dette problemet ikke er et klassisk envareflytproblem. Hvis vi kunne sett bort fra disse restriksjonene ville dette problemet hatt heltallsegenskapen, *integrality property*, som nevnt i kapittel (3.1.1). Fordi denne egenskapen ikke er oppfylt kan vi derfor ikke relaksere binærkravet.

Grunnen til at restriksjonene (5.12)-(5.14) bryter med denne heltallsegenskapen er fordi disse restriksjonene ved et minimeringsproblem fører til at en kan oppnå en bedre optimal løsning ved å sette en fraksjonell verdi på flytvariabelen. Dette kan vi for eksempel se ved å ta en nærmere titt på restriksjon (5.12). Den lineære versjonen ser som nevnt ut som følger:

$$z_i + KMTT_{ij} + LENGDE_j - z_j \leq M1_{ij}(1 - x_{ij})$$

Hvis vi har binærkrav vil denne restriksjonen se ut som følger:

$$\begin{aligned} x_{ij} = 1, \quad &\rightarrow \quad KMTT_{ij} + LENGDE_j \leq z_j \\ x_{ij} = 0, \quad &\rightarrow \quad KMTT_{ij} + LENGDE_j \leq M1_{ij} + z_j \end{aligned} \tag{5.15}$$

Hvis det derimot er tillatt med kontinuerlig verdi på flytvariabelen kan restriksjonene se ut som følger:

$$x_{ij} = \frac{1}{2}, \quad \rightarrow \quad KMTT_{ij} + LENGDE_j \leq \frac{M1_{ij}}{2} + z_j \tag{5.16}$$

Vi ser dermed at tellervariabelen kan settes til null selv om det er det er flyt mellom to noder. Dermed kan en unngå å telle opp kilometrene som kjøres ved å dele opp flyten i fraksjonelle verdier. Siden vi har et minimeringsproblem med målfunksjon som indirekte er avhengig av denne kilometertelleren vil dette være tilfelle her, det vil si at det blir ingen vedlikehold hvis ikke telleren teller opp. Hadde vi derimot hatt et

tilsvarende maksimeringsproblem ville ikke dette problemet oppstått. Dermed kunne vi relaksert binærkravet på flytvariablene i et envareflytproblem. Det er ikke sikkert at det å relaksere binærkravet fører til en lovlig løsning for et flervareflytproblem [Rardin, 1998].

Vedlikeholdkapasitet

For å håndtere at hver vedlikeholdsbase har begrenset kapasitet har vi defineret en node j for hvert tillatte vedlikehold. Restriksjon (5.6) sier at maksimalt tillatt flyt inn i hver slik vedlikeholdsnode er lik en. Dermed er restriksjonen om at hver vedlikeholdsbase har begrenset kapasitet blitt tatt hensyn til.

Overnattingskapasitet

Fordi vi bruker dagsløp som inndata til vår modell har vi ingen oversikt over ved hvilke stasjoner hvert tog befinner seg inni et dagsløp. Vi vil derfor ikke se på hensettingskapasitet på dagtid. Vi vil derimot modellere overnattingskapasiteten ved stasjonene på nattestid.

$$\sum_{\substack{i|SLUTTSTED_i = s \\ (T_i = t)}} \sum_{\substack{j \in O_i \\ (T_j = t(modTTOTAL)+1)}} x_{ij} - y_{st} = 0 \qquad \forall s \in S, t \in G \tag{5.17}$$

$$y_{st} \leq HKAP_s \qquad \forall s \in S, t \in G \tag{5.18}$$

$$y_{st} \geq 0 \, og \, heltall \qquad \forall s \in S, t \in G \tag{5.19}$$

Beskrankning (5.17) summerer antall tog som overnatter ved en stasjon s for en gitt dag t. Tilhørende dag for en node er definert som den dagen aktiviteten ved noden starter. Summen av alle tog som overnatter på en stasjon s på en gitt dag t, y_{st}, er summen av alle noder i som stopper på denne stasjonen, $SLUTTSTED_i = s$, gitt at toget ikke kjører videre før dagen etter. At utnoden j til node i ikke starter før dagen etter blir oppfylt ved at $T_i = t$ og $T_j = (t(modTTOTAL) + 1)$. (5.18) passer på at overnattingsvariabelen, y_{st}, ikke overskrider hensettingskapasiteten for stasjon s, $HKAP_s$. (5.19) setter heltallskrav på hensettingsvariabelen.

Problemet med å formulere restriksjonen på denne måten er at man antar at tomtogskjøringen mellom node i og node j blir gjennomført rett før node j starter. I noen tilfeller kan det bli mer optimalt å sette opp tomtoget på kvelden. Det vi si at toget da overnatter på startstasjonen til node j, $STARTSTED_j$, i stedet for å overnatte på sluttstasjonen til node i, $SLUTTSTED_i$. I andre tilfeller vil man kjøre tomtog både etter at et dagsløp er avsluttet og før det neste begynner. Det vil si at toget ikke overnatter på verken på stasjon $SLUTTSTED_i$ eller $STARTSTED_j$. Dette kan skyldes at overnattingskapasiteten på noen stasjoner er lav og det derfor vil lønne seg å sette toget til overnatting ved en annen stasjon med større kapasitet. Denne fleksibiliteten i tomtogskjøring har vi ikke tatt hensyn til i modellen vår.

Materiellтilgjengelighet

I inndataene til modellen vår er det gitt hvor mange tog som brukes av hver materielltype. Disse dataene tar dermed naturlig nok hensyn til at kapasiteten for hver materielltype ikke blir overskredet. Siden vi i vår modell har tillatt å dele opp en del av disse dagsløpene i mindre deler er det dermed ikke lenger gitt at denne materielltilgjengelighetskapasiteten er overholdt. I dette problemet har vi kun en togtype og dermed vil restriksjonen se ut som følger:

$$\sum_{i|T_i=T} \sum_{j|T_j=1} x_{ij} - w \leq 0 \tag{5.20}$$

$$w \leq MATKAP \tag{5.21}$$

$$w \geq 0 \, og \, heltall \tag{5.22}$$

(5.20) summerer all flyten i overgangen fra en søndag, t=T, til en mandag, t=1. Siden lengden på planperioden i dette tilfelle kun er på en uke, vil antall tog som trengs tilsvare summen av flyten ved et tilfeldig tidspunkt. Dette tidspunktet var her valgt til å være overgangen mellom søndag og mandag. (5.21) begrenser antall tog, w, slik at materielltilgjengeligheten, $MATKAP$, ikke blir overskredet. (5.22) gir ikke-negativitets og heltallskrav til materiellvariabelen w.

5.3.3 Anvendelse av EntypeProblemet

Denne modellen kan brukes til å løse en forenklet versjon av vedlikeholdsproblemet ved å bruke modellen sekvensielt for en togtype om gangen. Dette kan gjøres ved å inkludere alle nodene og dataene tilhørende den gjeldende togtypen, samt alle vedlikeholdsnodene der denne togtypen er tillatt. Tilsvarende inndata brukes for alle togtypene, bortsett fra at en da vil fjerne alle vedlikeholdsnodene som allerede er brukt av den(de) foregående togtypen(e). På tilsvarende måte må en redusere hensettingskapasiteten ved stasjonene. Denne sekvensen vil fortsette for hver togtype man ønsker å løse problemet for.

Denne sekvensielle løsningsmetoden gir ikke nødvendigvis en optimal løsning. En metode for å sjekke hvor god denne løsningen er kan gjøres ved løse problemet med varierende rekkefølge på togtypene. Løsningen kan variere avhengig av i hvilken rekkefølge togtypene blir optimert.

Fordi denne modellen kun gjelder for en uke kan det være vanskelig å få kjørt alle togsettene opp mot sin kilometergrense. Dette kan føre til antall vedlikehold kan bli høyere en det som en nødvendig med en lengre planleggingshorisont, se Flerukerproblemet i kapittel 5.6.

Uansett vil denne løsningen gi en øvre grense for målfunksjonsverdien for det komplette problemet.

5.4 FlertyperProblemet

Ved å utvide modellen til å gjelde flere materielltyper om gangen, må man legge til en indeks for togtypen i forhold til Entypeproblemet i kapittel 5.3. I tillegg har dagsløpsnodene en indeks for hvilken togtype som gjelder og vedlikeholdsnodene hvilke togtyper som kan entre noden.

5.5 Modellen for FlertyperProblemet

For Flertyperproblemet vil målfunksjonen og alle restriksjonene fra kapittel 5.3 gjelde. Den eneste forskjellen er at vi har en indeks k på variablene, nodene og kilometergrensen.

Målfunksjonen

$$min \sum_{k \in K} \sum_{j \in V^k} \sum_{i \in I_j} C^V x_{ij}^k + \sum_{k \in K} \sum_{j \in N^k} \sum_{i \in I_j} C^{TT} KMTT_{ij} x_{ij}^k \tag{5.23}$$

Flyt

$$\sum_{i \in I_j} x_{ij}^k - \sum_{i \in O_j} x_{ji}^k = 0 \qquad \forall k \in K, j \in N^k \tag{5.24}$$

$$\sum_{i \in I_j} x_{ij}^k = 1 \qquad \forall k \in K, j \in D^k \tag{5.25}$$

$$\sum_{k \in K} \sum_{i \in I_j} x_{ij}^k \leq 1 \qquad \forall j \in V^k \tag{5.26}$$

Vedlikehold

$$z_i - z_j + M1_{ij}^k x_{ij}^k \leq M1_{ij} - KMTT_{ij} - LENGDE_j \qquad \forall k \in K, j \in D^k, i \in D^k \cap I_j \tag{5.27}$$

$$z_i + M2_{ij} x_{ijt} \leq M2_{ij}^k - KMTT_{ij} - KMGRENSE \qquad \forall k \in K, j \in V^k, i \in D^k \cap I_j \tag{5.28}$$

$$M3_{ij} x_{ij} - z_i \leq M3_{ij} - KMTT_{ij} - LENGDE_j \qquad \forall k \in K, j \in D^k, i \in V^k \cap I_j \tag{5.29}$$

$$0 \leq z_j \leq KMGRENSE^k \qquad \forall k \in K, j \in D^k \tag{5.30}$$

$$x_{ij}^k \in \{0, 1\} \qquad \forall k \in K, j \in N^k, i \in I_j \tag{5.31}$$

Hensettingskapasitet

$$\sum_{k \in K} \sum_{\substack{i | SLUTTSTED_i = s \\ (T_i = t)}} \sum_{\substack{j \in O_i \\ (T_j = t (mod TTOTAL) + 1)}} x_{ij}^k - y_{st} = 0 \qquad \forall s \in S, t \in G \tag{5.32}$$

$$y_{st} \leq HKAP_s \qquad \forall s \in S, t \in G \tag{5.33}$$

$$y_{st} \geq 0 \, og \, heltall \qquad \forall s \in S, t \in G \tag{5.34}$$

Materielltilgjengelighet

$$\sum_{i | T_i = T} \sum_{j | T_j = 1} x_{ij}^k - w^k \leq 0 \qquad \forall k \in K \tag{5.35}$$

$$w^k \leq MATKAP^k \qquad \forall k \in K \tag{5.36}$$

$$w^k \geq 0 \, og \, heltall \qquad \forall k \in K \tag{5.37}$$

Alle restriksjonene i Flertyperproblemet tilsvarer restriksjonene for EnvareflytProblemet i kapittel 5.3. For dette problemet vil verdiene på $M1_{ij}$ og $M2_{ij}$ være avhengig av k, det vil si $M1_{ij}^k$ og $M2_{ij}^k$. Dette er fordi disse verdiene beregnes ut fra $KMGRENSE^k$ som er ulik for de forskjellige togtypene. Beregningen av disse verdiene vil være tilsvarende som for Entypeproblemet, se kapittel 5.3, og vi viser derfor ikke disse her.

5.5.1 Anvendelse av FlertyperProblemet

Fordi denne modellen løser problemet for alle togtypene samtidig vil denne kostnaden være mindre eller lik summen av kostnadene ved å løse problemet sekvensielt for hver togtype. Dette problemet vil dermed gi en strammere øvre grense enn Entypeproblemet for vedlikeholdsproblemet.

5.6 FlerukerProblemet

Vi ønsker å tillate flere uker i planperioden for å se hvordan dette påvirker vedlikeholdsplanleggingen.

I et slikt problem ønsker vi at koblingene mellom dagsløp skal være like hver uke. Dette gjelder for alle dagene i en uke bortsett fra koblinger mellom søndager og mandager hvor det er tillatt med variasjoner. Derimot kan koblingene mellom dagsløp og vedlikehold variere. Det vil si at hvis to dagsløpnoder i og j følger etter hverandre en uke, må de følge etter hverandre alle ukene. Den eneste tillatte variasjonen fra uke til uke er dermed plassering av vedlikeholdene. Dette kravet ble automatisk oppfylt for EnTypeProblemet og Flertypeproblemet der planperioden kun bestod av en uke. I dette problemet innfører vi derfor en indeks u som indikerer hvilken uke en node tilhører.

I tillegg har vi konstantene $PTOTAL$ som gir antall dager i planperioden og U som gir antall uker i planperioden. T er settet med dager i en grunnuke. Grunnuke er det mønsteret som vi ønsker skal gjenta seg i perioden, som i vårt tilfelle vil være en vanlig uke fra mandag til søndag. $TTOTAL$ er antallet dager i en grunnuke. Beslutningsvariablene x^k_{iju} og z_{iu} har også fått denne ekstra ukesindeksen.

5.6.1 Målfunksjonen

Målfunksjonen tilsvarer målfunksjonen (5.23) for Flertypeproblemet, bortsett fra den ekstra ukesindeksen u på flytvariablen, x^k_{iju}. Kostnaden blir gitt som gjennomsnittskostnad for en grunnuke. Dette fordi det da er lettere å sammenligne kostnad med de to foregående modellene.

$$min \qquad \frac{1}{U}(\sum_{u\in U}\sum_{k\in K}(\sum_{j\in V}\sum_{i\in I_j} C^V x^k_{iju} + \sum_{j\in N}\sum_{i\in I_j} C^{TT} KMTT_{ij} x^k_{iju}))$$

5.6.2 Beskrankninger

Flytrestriksjonen

$$\sum_{i\in I_j} x^k_{iju} - \sum_{i\in O_j} x^k_{iju} = 0 \qquad \forall u \in U, k \in K, j \in N^k \tag{5.38}$$

$$\sum_{i\in I_j} x^k_{iju} = 1 \qquad \forall u \in U, k \in K, j \in D^k \tag{5.39}$$

$$\sum_{k\in K}\sum_{i\in I_j} x^k_{iju} \leq 1 \qquad \forall u \in U, j \in V^k \tag{5.40}$$

Vedlikehold

$$z_{iu} - z_{ju} + M1_{ij} x^k_{iju} \leq M1_{ij} - KMTT_{ij} - LENGDE_j \qquad \forall u \in U, k \in K, j \in D^k, i \in D^k \cap I_j \tag{5.41}$$

$$z_{iu} + M2_{ij} x^k_{iju} \leq M2_{ij} - KMTT_{ij} - KMGRENSE \qquad \forall u \in U, k \in K, j \in V^k, i \in D^k \cap I_j \tag{5.42}$$

$$M3_{ij} x^k_{iju} - z_{iu} \leq M3_{ij} - KMTT_{ij} - LENGDE_j \qquad \forall u \in U, k \in K, j \in D^k, i \in V^k \cap I_j \tag{5.43}$$

$$0 \leq z_{ju} \leq KMGRENSE^k \qquad \forall u \in U, k \in K, j \in D^k \tag{5.44}$$

$$x^k_{iju} \in \{0,1\} \qquad \forall u \in U, k \in K, j \in N^k, i \in I_j \tag{5.45}$$

Hensettingskapasitet

$$\sum_{k \in K} \sum_{\substack{i | SLUTTSTED_i = s \\ (T_i = t)}} \sum_{\substack{j \in O_i \\ (T_j = t (mod TTOTAL) + 1)}} x^k_{iju} - y_{stu} = 0 \qquad \forall s \in S, t \in G, u \in U \tag{5.46}$$

$$y_{stu} \leq HKAP_s \qquad \forall s \in S, t \in G, u \in U \tag{5.47}$$

$$y_{stu} \geq 0 \, og \, heltall \qquad \forall s \in S, t \in G, u \in U \tag{5.48}$$

Som tidligere nevnt har vi antatt i vår modell at tomtogskjøring til et dagsløp blir gjennomført rett før et dagsløp blir gjennomført. Selv om det er akkurat de samme dagsløpene som skal dekkes hver uke, må en likevel sjekke hensettingskapasiteten for hver uke. Dette kommer av det er variasjoner i vedlikehold. Dermed kan et tog som dekker samme dagsløp i flere etterfølgende uker overnatte på stasjonen den ene uken og på vedlikeholdsbasen den neste uken.

Materielltilgjengelighet

$$\sum_{i | T_i = T} \sum_{j | T_j = 1} x^k_{ij11} - w^k \leq 0 \tag{5.49}$$

$$w^k \leq MATKAP^k \qquad \forall k \in K \tag{5.50}$$

$$w^k \geq 0 \, og \, heltall \qquad \forall k \in K \tag{5.51}$$

For å sjekke antall tog som trengs er det i likhet med tidligere tilstrekkelig å sjekke hvor mange tog som trengs ved et vilkårlig tidspunkt. Om et tog dekker forskjellige dagsløp hver uke, vet vi at et eller flere andre tog dekker de dagsløpene som dette toget ikke dekker den gjeldende uken.

Like Uker

I motsett til EntypeProblemet og Flertyperproblemet vil ikke kravet om like uker automatisk blir oppfylt. Derfor innfører vi følgende restriksjon:

$$\begin{aligned} &(x^k_{iju} + \frac{1}{2} \sum_{l \in V^k \cap O_i \cap I_j} (x^k_{ilu} + x^k_{lju})) - \\ &(x^k_{ij(u(modU)+1)} + \frac{1}{2} \sum_{l \in V^k \cap O_i \cap I_j} (x^k_{il(u(modU)+1)} + x^k_{lj(u(modU)+1)})) = 0 \\ &\qquad \forall k \in K, u \in U, j \in D^k, i \in D^k \cap I_j, \end{aligned} \tag{5.52}$$

På grunn av restriksjon (5.38) ser vi at maksimalt et av uttrykkene $\frac{1}{2}\sum_{l\in V^k\cap O_i\cap I_j}(x^k_{ilu}+x^k_{lju})$) og x^k_{iju} vil være lik en. Dette betyr at enten er node i og node j koblet direkte sammen eller så ligger det et vedlikehold l mellom i og j. Kravet om at vi ønsker like uker innebærer at hvis to dagsløpsnoder kobles sammen en uke, ønsker vi at disse to dagsløpsnodene skal kobles sammen de resterende ukene i planperioden. Dette er uavhengig om det er et vedlikehold mellom dagsløpsnodene. Dette betyr at hvis dagsløpsnodene i og j kobles direkte en uke, kan de kobles med et vedlikehold mellom seg neste uke, for å så kobles sammen uten vedlikehold den tredje uken. Derfor vil enten uttrykket $(x^k_{ij(u(modU)+1)}$ eller $\frac{1}{2}\sum_{l\in V^k\cap O_i\cap I_j}(x^k_{il(u(modU)+1)}+x^k_{lj(u(modU)+1)})$) bli lik en dersom node i og j kobles sammen i en uke u og dermed sørge for at disse to nodene også blir koblet sammen, med eller uten vedlikehold, også uken etter.

5.6.3 Anvendelse av FlerUkerProblemet

I dette problemet ser vi på en planperiode på flere uker modellen håndterer flere togtyper samtidig. Dette gjør at vi kan øke fleksibiliteten ved at plassering av vedlikehold kan variere fra uke til uke. Dette gjør at togene har mulighet til å bedre fylle opp sin kilometergrense. Dermed vil kostnaden per uke for denne modellen maksimalt bli like stor som for Flertyperproblemet.

5.7 Set partitioning problem med vedlikehold

I denne problemformuleringen vil vi ta i bruk kolonner. Denne problemformuleringen er ikke fullt utarbeidet, men kan sees på som en start til et mulig videre arbeid.

5.7.1 Antagelser

Vi har i dette forslaget til en løsning sett bort i fra hensettingskapasiteten.

Hver av kolonnene vil innholde et eller flere dagsløp som er lovlig å koble sammen. For hver kolonne er det tilordnet et vedlikehold. Dermed vil hver kolonne starte og slutte ved en vedlikeholdsbase.

I problemet vårt kan hvert tog vedlikeholdes ved forskjellige vedlikeholdsbaser. Vi har i hver vedlikeholdsbase flere skift med mulighet for et eller flere vedlikehold. Hvert av disse mulige vedlikeholdene kan håndtere en eller flere togtyper, men kun et vedlikehold om gangen. Vi vil her se på alle mulige vedlikehold av samme togtype som et *vedlikeholdsdepot*. Dermed vil en del av vedlikeholdene være like for flere av vedlikeholdsdepotene. Det er imidlertid kun lov til å bruke maksimalt et av disse like vedlikeholdene. Dermed må vi innføre en fellesrestriksjon for hvert skift som gjelder for alle vedlikeholdsdepotene. En kan dermed dele kolonnene i subsett ut i fra hvilket vedlikehold de tilhører. Høyst en kolonne i hvert av disse subsettene vil dermed kunne være med i løsningen.

Kostnaden for hver av disse kolonnene vil bestå av den tomtogskjøringen som er nødvendig for å gjennomføre dagsløpene i kolonnen. Dermed blir problematikken ved at hvert tog kan bli vedlikeholdt ved vedlikeholdsbaser som har ulik geografisk plassering håndtert i kostnaden for hver kolonne. Denne kostnaden vil dermed inkludere avstanden fra vedlikholdsbasen og til startstasjonen i første dagsløp i kolonnen.

5.7.2 Generering av alle lovlige kolonner

For å løse problemet ved set partitioning må en først generere en mengde kolonner som representerer lovlige delløsninger for det opprinnelige problemet. En mulig framgangsmåte for å lage disse kolonnene kan være å starte ved å ta utgangspunkt i et mulig vedlikehold. Ut fra dette vedlikeholdsskiftet kan en generere en sekvens av lovlige dagsløp. Disse kolonnene vil dermed bli generert slik at de tar hensyn til kilometerrestriksjonen. Hver kolonne vil dermed starte og slutte i vedlikeholdsbase for den gitte togtypen.

Dette kan gjøres ved at en har en ordnet sekvens med dagsløp: $(1, 2, 3, 4, 5, 6)$. Ut fra denne sekvensen kan en for eksempel generere tre lovlige sekvenser av dagsløp, $(1, 2, 3)$, $(2, 3, 4)$ og $(3, 4, 5, 6)$. Ingen av disse kolonnene vil ha en lengde som overskrider dem maksimale kilometergrensen. Hver av disse sekvensene vil representere en kolonne, x_i.

Hver kolonne vil dermed få en kostnad C_i^V og C_i^{TT}. Kostnaden C_i^V er kostnaden per vedlikehold og C_i^{TT} er kostnaden av tomtogskjøringen som er inkludert i kolonne i. Denne tomtogskjøringen inkludere både kjøring mellom alle dagsløpene og til og fra vedlikeholdsbasen. Kostnaden av hvert dagsløp er ikke inkludert fordi denne vil være lik for alle mulige løsninger av problemet. Disse to kostnadene til sammen kan kalles C_i.

5.8 Notasjon

Følgende data vil bli gitt inn i modellen:

Indekser

- i kolonne
- j dagsløp
- d dag i planleggingshorionten
- v vedlikeholdsskift
- b vedlikeholdsbase

Konstanter

- I Antall kolonner for planleggingsperioden
- J Antall dagsløp i planleggingsperioden
- D Antall dag i planleggingen
- T Antall vedlikeholdsskift
- V Antall vedlikeholdsbaser
- C_i Kostnaden for kolonne i

Mengder

$$A_{ij} = \begin{cases} 1 & \text{hvis subsett } i \text{ dekker dagsløp } j \\ 0 & \text{ellers} \end{cases}$$

$$B_{ivd} = \begin{cases} 1 & \text{hvis subsett } i \text{ tilhører vedlikehold } v \text{ på dag } d \\ 0 & \text{ellers} \end{cases}$$

Beslutningsvariabel

$$x_i = \begin{cases} 1 & \text{hvis subsett } i \text{ er en del av løsningen} \\ 0 & \text{ellers} \end{cases}$$

5.8.1 Modellen

$$min \sum_{i=1}^{I} C_i^V x_i + \sum_{i=1}^{I} C_i^{TT} x_i \tag{5.53}$$

$$\sum_{i=1}^{I} A_{ij} x_i = 1 \qquad \forall j \in \{1, J\} \tag{5.54}$$

$$\sum_{i=1}^{I} B_{ivd} x_i \leq 1 \qquad \forall v \in \{1, V\}, \forall d \in \{1, D\} \tag{5.55}$$

$$x_i = \{0, 1\} \qquad \forall i \in \{1, I\} \tag{5.56}$$

Målfunksjonen (5.53) minimerer kostnaden for de kolonnene som inngår i løsningen av problemet. Restriksjonen (5.54) er set partitioning delen av problemet. Dette restriksjonssettet sørger for at utvalget av kolonner som er med i løsningen dekker hvert dagsløp j akkurat en gang. Restriksjonen (5.55) sørger for at hvert mulig vedlikehold v for hver dag d dekkes maksimalt en gang. Restriksjonen (5.56) er binærkravet til beslutningsvariabelen x_i.

I tillegg til disse restriksjonene må vi ha en restriksjon som sikrer at det finnes koblinger mellom kolonnene. Dette betyr at det må finnes et ledig tog i det man skal starte en kolonne, det vil si at en starter et vedlikehold. Denne restriksjonen kan sees på som en lagerbeskrankning. For hvert vedlikeholdsbase kan man definere gitte tidspunkt t der et tillatt at kolonne en kan starte. Disse tidspunktene kan være starten av hvert vedlikehold. For hver kolonne som må en derfor sjekke om det er et ledig tog.

$$TILGJENGELIG(t) = TILGJENGELIG(t-1) - STARTET(t-1) + STOPPET(t)$$

Antall tog som er tilgjengelig ved tidspunkt t vil være lik antall tog som er tilgjengelig ved tidspunkt $(t-1)$ minus antall tog som startet ved vedlikeholdsbasen ved tid $(t-1)$ pluss antall tog som har ankommet vedlikeholdsbasen i tiden mellom t og $(t-1)$. En slik formulering av lagerbeskrankningen betyr at det kun starter en kolonne i hvert tidspunkt. Dette vil imidlertidig ikke være tilfelle. Vi har ikke lagt vekt på å gi en fullstendig formulering her, bare noen tanker om hvordan en kan starte en formulering.

Kapittel 6

Implementering og resultater

Vi har implementert de tre modellene, EntypeProblemet, FlertyperProblemet og FlerukerProblemet fra kapittel 5. Vi har ved testing vekslet mellom ulike egenskaper hentet fra disse problemene. I dette kapitlet vil vi først forklare hvordan vi har implementert modellene og deretter vil vi presentere resultatene.

6.1 Implementering

Vi har benyttet en inkrementell implementeringsstrategi ved at vi startet med en enkel basismodell og gradvis utvidet denne til den tilfredstilte kravene til FlerukerProblemet fra kapittel 5. Koden til alle tre problemene har vi lagt i en Mosel-fil, se vedlegg B. Et eksempel på mulig inndata ligger i vedlegg C.

Her vil vi først si noen ord om programvaren vi har brukt og deretter se på hvordan vi har valgt å implementere modellene.

6.1.1 Programvare

Vi har brukt Xpress-IVE for å implementere modellene våre. Grunnen til dette er at denne programvaren ble det gitt undervisning i i faget Optimeringsmetoder. Samtidig er det dette programmet vi har tilgang til på studentdatasalene. Vi har brukt Xpress-IVE Version 1.14.33, Xpress Mosel Version 1.2.4 og Xpress Optimizer 14.24. Testingen har foregått på en PC med P4 2.4GHz og 256 MB minne. Kildekoden for Flertypeproblemet ligger i vedlegg B.

6.1.2 Implementerings detaljer

Vi vil her beskrive og forklare noen implementeringstekniske valg vi har gjort.

Parametre

For å velge hvilken funksjonalitet dem implementerte modellen skal ha, må man sette noen parametre først i modellkoden. Disse er vist i tabell 6.1. Entypeproblemet og Flertypeproblemet blir gitt ut fra hvilken inndatasett som velges. Ved valg av TTOTAL,

Tabell 6.1: Parametre i implementeringen

Parameter	Forklaring
TTOTAL	Lengden på en grunnuke
PTOTAL	Lengden på planperioden
SETT1	Valg av inndatasett
LIKEUKER	1 = Like uker 0 = Forskjellige uker
TOMTOG	1 = Kjører tomtog 0 = Kjører ikke tomtog

PTOTAL og LIKEUKER kan en bestemme om en skal kjøre et Flerukerproblem. Tabellen forklarer hvordan disse parametrene settes. I tillegg kan en velge om det skal være tillatt å kjøre tomtog eller ikke.

Inndata

Vi har laget flere ulike datasett og et av disse er presentert i vedlegg C. Datasettene vi har brukt til testing er listet opp i tabell 6.2. Datasettene er laget på bakgrunn av materiellturneringsplanen 153.2. Denne planen inneholder alle rutene med tilhørende materiell som skal kjøres fra januar til juni 2005. Hvert av datasettene våre inneholder hele eller deler av materiellturneringsplanen. Denne planen består av en del sykler. Vi har laget datasettene våre slik at vi inkluderer hele sykler fra planen. Dette betyr at alle dagsløp som er knyttet sammen i planen er med i inndatasettet. Grunnen til dette er at da vet vi at det finnes en lovlig løsning av datasettet og for enklere å kunne sammenlikne løsningen vår med planen til NSB.

Vi har også foretatt noen beregninger selv. Siden vi ikke har hatt noen oversikt over avstandene i kilometer og tid mellom alle stasjonene, har vi i stedet anslått rimelige verdier. Dette kan være en årsak til at løsningen vår nok vil avvike litt fra løsningene til NSBs plan. På grunn av dette har vi heller ikke vektlagt å sammenlikne antall tomtogskilometer fra løsningene våre opp mot planen til NSB.

Vi har brukt feil kilometergrense for togtype 69 i inndataene våre i forhold til det NSB bruker i planen sin. Fra 2005 er kilometergrensen for togtype 69 blitt endret til 10.000km fra 6.000km. Derfor vil antallet vedlikehold ved testing av type 69 ofte være høyere enn i planen til NSB. For å sjekke om løsningen gir en forbedring må vi sjekke med en teoretisk verdi.

Nedre grense for problemet

Xpress bruker Branch and Bound som løsningsstrategi, og lineærrelaksering til å finne en nedre grense på problemet. Når problemet vårt blir LP-relaksert, vil resultatet bli null vedlikehold på grunn av at kilometertelleren blir null, se kapittel 5.3. Vi har derfor valgt å legge inn en ekstra restriksjon som tvinger målfunksjonen over et gitt minstemål. Dette minstemålet er gitt ved lengden på alle dagsløp delt på kilometergrensa. Dette

Tabell 6.2: Egenskaper for inndatasettene

Navn	DLnoder	VLnoder	Egenskap	Togtype
typer1	14	14	Uten tomtog	Type72
typer2	28	14	Uten tomtog	Type72
typer3	28	14	Med tomtog	Type72
typer4	28	14	Uten tomtog	Type72
typer5	42	42	Med tomtog	Type69,72
typer6	28	36	Uten tomtog	Type69,72
typer7	42	14	Uten tomtog	Type72
typer8	56	14	Uten tomtog	Type72

vil gi en nedre grense på problemet som vil ligge nærme opptil den optimale løsningen. Dette kravet for FlertypeProblemet vil da være gitt ved restriksjonen:

$$Objektfunksjon \geq U * \sum_{k \in K} \sum_{j \in N} \frac{LENGDE_j}{KMGRENSE^k} * NODETYPE_j^k \tag{6.1}$$

Tomtogskjøring

I utgangspunktet er det tillatt med tomtogskjøring mellom alle noder om ikke annet er uttrykt eksplisitt.

Vi har i tillegg også inkludert et valg om at det bare skal være lov til å kjøre tomtog til og fra vedlikeholdsbasene. Dette betyr at bare dagsløpsnoder som stopper og starter på samme sted kan kobles sammen. Dette krever en annen type inndata. I dette tilfellet må inndataen inkludere tomtog slik at balansen ved hver stasjon er riktig. Det betyr at hvis et dagsløp starter ved en stasjon, må et annet dagsløp ha blitt avsluttet på samme stasjon før dette startet.

Grunnen til å inkluderer dette valget er for å se på hva muligheten for tomtogskjøring har å si for problemstørrelsen og løsningstid. Når tomtog er tillatt blir det flere lovlige inn og utnoder per node, og dermed vil problemet få flere beslutningsvariabler.

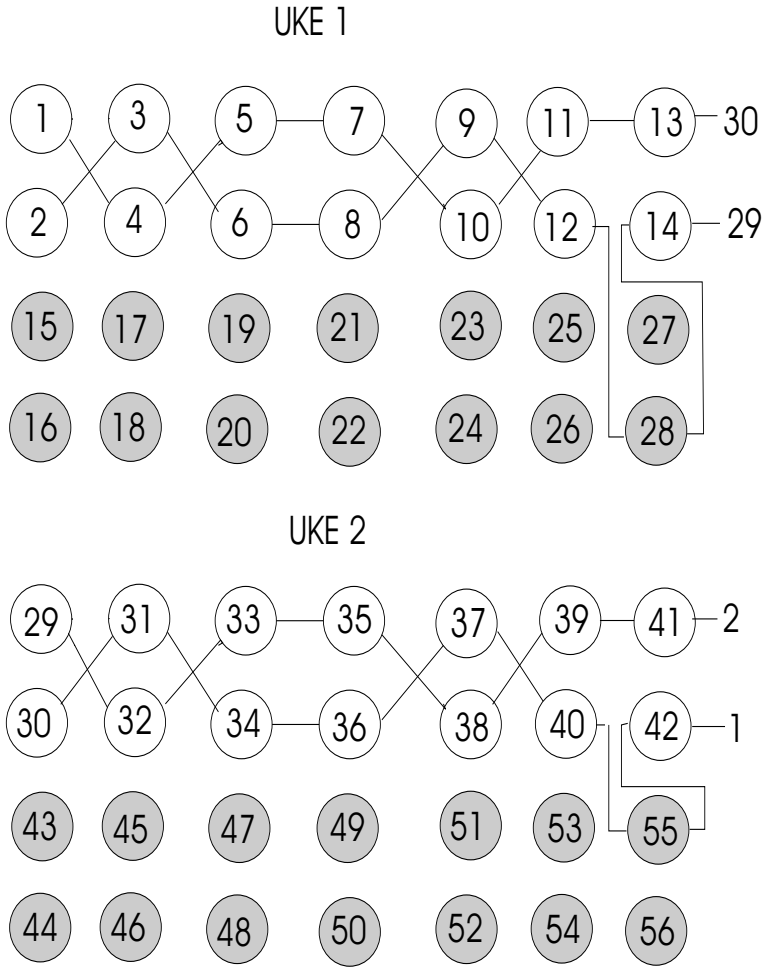
Overnattingskapasitet

Vi har i implementeringen ikke tatt hensyn til overnattingskapasiteten. Grunnen til dette er at da vi startet med implementeringen fant vi etter hvert ut at vi ikke ville klare å løse problemet for store datasett. Dermed vil det ikke være like relevant å se på overnattingskapasiteten, og vi valgte å ikke bruke tid på å implementere denne. I stedet teller vi opp hvor mange tog som slutter på hver stasjon hver dag på slutten av dagen, og sammenlikner med hensettingskapasiteten for stasjonen.

6.2 Resultater

Løsningstiden for modellen øker drastisk når størrelsen på problemet øker. En grunn til dette er at vi har et *blandet heltallsproblem* med binære og kontinuerlige beslutningsvariable. Strukturen på datasettene er slik at når størrelsen på datasettet øker, vil antallet dagsløp som skal dekkes øke og antallet lovlige innoder for hver node vil øke. Dette gjør at antallet beslutningsvariabler vil øke. Dette gjør at problemet blir vanskeligere å løse. De binære beslutningsvariablene bidrar sterkt til denne økningen i løsningstid.

Ved testingen av store datasett stoppet vi kjøringen før den hadde gitt en optimal løsning. Xpress fant imidlertid ofte en eller flere lovlige løsninger. For å gi en indikasjon på hvor god denne lovlige løsningen er, kan vi sammenligne med den nedre grensen og planen til NSB. For de problemene der vi stoppet kjøringen sier løsningstiden i tabellene når vi stoppet programmet. Denne verdien sier dermed ikke noe om når den eventuelle lovlige løsningen blir funnet.



Figur 6.1: Sammenkobling av to dagsløp i to uker

6.2.1 Entypeproblemet

En løsning for et Entypeproblem er gitt figur 6.1 og i tabell 6.3. Inndataen til problemet er gitt i tabell C.1, se vedlegg C. Figur 6.1 er en løsning på et problem som består av en togtype, to uker og der vi krever like uker. Figuren viser koblingene mellom dagsløpsnodene (de hvite) og vedlikeholdsnodene (de grå). De øverste nodene tilhører den første uken og de nederste nodene den andre uken.

Av tabell 6.3 ser vi at løsningen av problemet for to uker blir lik uansett om en krever like uker eller ikke. Dette skyldes sannsynligvis at datasettet er så lite. Vi ser at denne løsningen gir en fordobling av antall vedlikehold og antall tomtogskilometer i forhold til det samme problemet for en uke.

Tabell 6.3: Løsning for EntypeProblemet med datasett typer 1

Modell	**En type**	**En type**	**En type**
Ant uker	en uke	to uker	to uker
Datasett	typer1	typer1	typer1
DLnoder/VLHnoder	14/14	28/28	28/28
Like uker	-	ja	nei
TT tillatt	ja	ja	ja
Ant variable modell/presolved	117/106	234/190	234/221
Ant restriksjoner modell/presolved	157/145	336/289	312/300
Løsning	optimal	optimal	optimal
Løsningstid	0,2s	0,7s	0,6s
LP-relaksert løsning	0,999	1,71	1,71
Ant vedlikehold	1	2	2
Ant Vedlikehold hos NSB	1	2	2
Ant TT-km	325,1km	650,2km	650,2km

Presolved er antall variable/restriksjoner i modellen etter at Xpress har utført presolved. Modell er antall variable som modellen er formulert med i utgangspunktet.

Tabell 6.4 viser løsning av et Envareproblem med datasett typer3. Dette datasettet inkluderer tomtog. Vi har testet dette datasettet det er lov og ulovlig med tomtogskjøring. Vi ser at antallet beslutningsvariable blir lavere når det ikke er tillatt med tomtog. Dette skyldes at uten tomtogskjøring kan kun dagsløpsnoder som slutter og starter på samme stasjon kobles sammen. Tilsvarende blir det færre restriksjoner uten tomtogskjøring. I tabellene 6.3-6.7 er løsningene for entypeproblemene vist. Vi ser at løsningstiden for de minste modellene er liten, og at det er kortere løsningstid når man bare løser for en uke kontra to uker.

I tabell 6.5 viser vi resultatet av testene av to datasett av lik størrelse. Denne tabellen viser at Xpress kun klarte å løse flerukerproblemet for det ene settet. Dette viser at

modellen vår er svært sensitiv for strukturen på inndataene.

Ut fra disse tabellene ser vi at løsningstiden for de minste modellene er liten, og at det er kortere løsningstid når man bare løser for en uke kontra to uker.

6.2.2 FlerTyperProblemet

Tabellene 6.8 og 6.9 gir løsninger på to Flertyperproblem med relativt små inndatasett. Inndataene til disse to testene er datasettene typer6 og typer5. Vi ser at Xpress klarer å finne en lovlig løsning når datasettene er relativt små.

Vi har forsøkt å teste modellen vår med datasettet som gjelder for hele problemet til NSB, se tabell 6.10. Dette problemet er veldig stort og maskinen klarte bare å finne LP-relakseringen. Dermed forsøkte vi i stedet å løse problemet for hver av de tre togtypene for seg. Antall variable og restriksjoner for disse problemene er også presentert i tabell 6.10. Resultatet fra testene sier at Xpress finner en optimal løsning for typer 69-2 og en lovlig løsning for typer-72. For type 72 har vi sjekket at kjørte kilometer tomtog er riktig. Når vi ser nærmere på løsningene oppdager vi imidlertid at løsningen for typer 69-2 er feil. Dette ser vi ved at antall tomtogskilometer blir oppgitt til å være null. Går vi inn og sjekker koblingene av dagsløp i løsningen, ser vi at den inneholder koblinger mellom dagsløp som slutter og starter på ulike stasjoner. Slike koblinger krever tomtogskjøring, og vi vet dermed at denne løsningen må være feil. Dette datasettet har mange korte dagsløp, og det blir kun 2 vedlikehold per uke.

Vi har ikke funnet noen god forklaringen på denne feilen. Det vi vet er at det er flere beslutningsvariable i problemet for 69-2 enn for type 72. Dette skyldes at togtype 69 kan vedlikeholdes på begge vedlikeholdsbasene. For å teste om dette er grunnen til at programmet kjørte feil testet vi med et alternativt inndatasett. Dette datasettet har tilsvarende dagsløp, men kan bare vedlikeholdes på en av vedlikeholdsbasene, nemlig på Filipstad. Denne løsningen gir oss tilsvarende feil selv om dette problemet hadde færre variable og restriksjoner enn problemet for togtype 72. En av forklaringene på at modellen kjører feil kan være at datasettet for typer 72 og typer 69 er forskjellige. Dagsløpene med type 72 er betydelig lengre enn dagsløpene til type 69-2. Fordi vi startet testingen av modellen vår med togtype 72, oppdaget vi denne feilen med type 69 sent i arbeidet. Vi har derfor ikke hatt tid til å finne en sikker årsak til denne feilen.

6.2.3 FlereukerProblemet

Vi forsøkte å løse Flerukerproblemet både med og uten kravet med like uker. Vi forventet at løsningstiden for to like uker ville være kortere enn løsningstiden for to ulike uker. Grunnen til dette er at løsningsrommet blir betraktelig redusert, det vil si at det er færre frihetsgrader for variablene. Resultatet etter kjøringene er presentert i tabellene 6.3 - 6.8. Man kan se en forskjell på antall variabler og restriksjoner etter at Xpress har foretatt presolve på modeller med kravet til like uker og ikke. Presolve betyr at programmet har fjernet restriksjoner og variabler som er redundante. Med kravet til like uker blir det færre variabler og i flere tilfeller færre restriksjoner i problemet, og generelt blir det også kortere løsningstid.

Det var bare et par av testkjøringene som fant optimal løsning når planperioden var to uker. For det minste datasettet i tabell 6.3 ser vi at løsningstiden for problemene er

Tabell 6.4: Løsning for EntypeProblemet med datasett typer 3

	1	2	3	4	5	6
Modell	**Entype**	**Entype**	**Entype**	**Entype**	**Entype**	**Entype**
Ant uker	en uke	to uker	to uker	en uke	to uker	to uker
Datasett	typer3	typer3	typer3	typer3	typer3	typer3
DLnoder/ VLHnoder	28/14	56/28	56/28	28/14	56/28	56/28
Like uker	-	ja	nei	-	ja	nei
TT tillatt	nei	nei	nei	ja	ja	ja
Ant variable modell/presolved	216/193	432/334	432/408	265/246	530/413	530/510
Ant restriksjoner modell/presolved	272/245	598/489	542/515	321/298	737/617	640/617
Løsning	optimal	lovlig	lovlig	optimal	lovlig	lovlig
Løsningstid	85,1s	11931,6s	9359,3s	32,9s	8780,6s	8246,6s
LP-relaksert løsning	2,40	4,80	4,80	2,35	4,70	4,70
Ant vedlikehold	3	6	5	3	6	5
Ant Vedlikehold hos NSB	3	6	6	3	6	6
Ant TT-km	80	160	226,4	80	160	84

like, men dette kan skyldes at datasettet er så lite og at det å tillate ulike uker ikke ga store forskjeller på antall restriksjoner og variabler. På datasettet typer 2 i tabell 6.5 ser vi en større forskjell på løsningstiden med og uten kravet til like uker. På bakgrunn av kun denne testen kan vi ikke trekke en generell konklusjon, men det ser ut som om løsningstiden er kortere med kravet til like sykler.

6.2.4 Tomtogskjøring

Vi ønsket å teste hvordan tomtogskjøring ville påvirke løsningstiden og størrelsen på problemene. Tabellene 6.4 og 6.9 viser løsningene av slike tester. Datasettene i disse testene inkluderer dagsløpene med den tomtogskjøringen NSB planlegger å bruke, se tabell 6.2. At en ikke tillater tomtogkjøring betyr at det kun er lov til koble sammen dagsløp som slutter og starter ved samme stasjon. Dermed er det kun lov til å sette opp tomtogskjøring til og fra vedlikeholdsbasene. Det blir da færre mulige koblinger mellom nodene. Det fører til at det er færre variable og restriksjoner.

Erlebach et al. [2004] sier at en modell, uten krav til vedlikehold, som tillater tomtogskjøring har lenger løsningstid enn en modell som ikke tillater dette. Vi forventet derfor at problemet uten tomtogskjøring ville være raskere å løse enn problemet med tomgtogskjøring. Vi fikk likevel det overraskende resultatet at løsningstiden ble lenger når vi ikke tillater tomtog, se tabell 6.4, nummer 1 og 4. Vi ser imidlertidig at vi klarte

Tabell 6.5: Løsning for EntypeProblemet. To datasett med like mange noder.

	1	2	3	4	5	6
Modell	**Entype**	**Entype**	**Entype**	**Entype**	**Entype**	**Entype**
Ant uker	en uke	to uker	to uker	en uke	to uker	to uker
Datasett	typer2	typer2	typer2	typer4	typer4	typer4
DLnoder/ VLHnoder	28/14	56/28	56/28	28/14	56/28	56/28
Like uker	-	ja	nei	-	ja	nei
TT tillatt	ja	ja	ja	ja	ja	ja
Ant variable modell/presolved	273/258	546/443	546/529	267/248	534/417	534/514
Ant restriksjoner modell/presolved	327/311	748/636	652/636	323/300	741/621	644/621
Løsning	optimal	optimal	optimal	optimal	lovlig	lovlig
Løsningstid	2,0s	21,7s	64,6s	2,2s	12128,9s	4134,0s
LP-relaksert løsning	2,06	4,12	4,12	2,35	4,70	4,70
Ant vedlikehold	3	5	5	3	6	6
Ant Vedlikehold hos NSB	3	6	6	3	6	6
Ant TT-km	862,4	1534,8	1534,8	478,4	956,8	992

Programmet låste seg når Envareproblemet med datasettet typer3c og to ulike uker ble løst.

å finne en lovlig løsning med at lavere antall tomtogskilometer med tomtogskjøring enn uten, se kolonne 3 og 6. Dette resultatet var forventet fordi et problem som tillater tomtogskjøring har større fleksibilitet.

I tabell 6.9 ser vi at vi kun har fått en lovlig løsning av problemet. I denne testingen fant vi den lovlige løsningen mye raskere for problemet der tomtogskjøring ikke var tillatt. Dette framgår ikke av tabellen. Vi ser imidlertid at antall tomtogskilometer er lavere for problemet uten tomtogskjøring. Vi vet på forhånd at alle koblinger som er lovlige for et problem uten tomtogskjøring også er lovlige for et problem med tomtogskjøringen. Dette skyldes dermed at modellen ikke har kjørt lenge nok.

Erlebach et al. [2004] sier at vedlikehold gjør modellen kompleks og NP-hardt, både om tomtogskjøring er tillatt eller ikke, så derfor vil det nok avhenge av inndata hvilket av problemene som løses best.

6.3 Diskusjon

For togtype 72 har vi funnet en løsning med færre vedlikehold enn i planen til NSB. Dermed gir modellen vår en indikasjon på at det vil være mulig å redusere antall vedlikehold ved bruk av et beslutningsstøttesystem. I tillegg til at vår modell har redusert

Tabell 6.6: Løsning for EntypeProblemet med datasett typer 7

	1	2	3
Modell	**En type**	**En type**	**En type**
Ant uker	en uke	to uker	to uker
Datasett	typer7	typer7	typer7
DLnoder/VLHnoder	42/14	84/28	84/28
Like uker	-	ja	nei
TT tillatt	ja	ja	ja
Ant variable modell/presolved	496/473	992/749	992/966
Ant restriksjoner modell/presolved	562/539	1339/1099	1122/1099
Løsning	optimal	lovlig	lovlig
Løsningstid	39,5s	12029,7s	10638,2s
LP-relaksert løsning	3,20	6,41	6,41
Ant vedlikehold	4	8	8
Ant Vedlikehold hos NSB	4	8	8
Ant TT-km	710,5	1421,0	2046,6

antall vedlikehold, vil bruk av en slik modell kunne være svært tidsbesparende i en planleggingsprosess. Svakheten med denne løsningen er at den er uavhengig av de andre togtypene. Vi vet dermed ikke hvordan denne løsningen vil påvirke planleggingen av de resterende togtypene.

De fleste av datasettene våre er laget med togtype 72 og vi har brukt disse under utarbeidelsen av modellen. I ettertid ser vi at dette har vært en svakhet, siden modellen for type 69-2 kjører feil. Dagsløpslengdene for denne togtypen er korte, og NSB har kun to vedlikehold per uke for denne togtypen. Dermed er det lett å tro at det i utgangspunktet ikke er mye å spare på å inkludere denne togtypen i et beslutningsstøttesystem. En vet imidlertidig ikke hvordan denne løsningen vil påvirke vedlikeholdsplanleggingen av de andre togtypene, så vi kan ikke uten videre trekke en slik konklusjon.

Vi har underveis i testingen fått indikasjoner på at modellen er svært følsom på strukturen til inndataene. Tabell 6.5 gir resultat av noen tester på to datasett av lik størrelse. Vi har utført nøyaktig de sammen testene for begge datasettene. Likevel ser vi at Xpress kun klarte å løse det ene settet til optimal verdi, mens vi har stoppet det andre.

Når størrelsen på problemene øker, klarer ikke Xpress å løse problemet til optimalitet. Vi har likevel for en del av disse problemene fått en lovlig løsning. Disse løsningene har gitt oss et likt eller lavere antall vedlikehold enn det som NSB planlegger. For å kunne foreta en bedre sammenlikning av løsningen vår opp mot planen til NSB, burde vi ha lagt mer vekt på sammenligning av antall tomtogskilometer. Dette har vi

Tabell 6.7: Løsning for entypeproblemet med inndatasett typer 8

	1	2	3
Modell	**En type**	**En type**	**En type**
Ant uker	en uke	to uker	to uker
Datasett	typer8	typer8	typer8
DLnoder/VLHnoder	56/14	112/28	112/28
Like uker	-	ja	nei
TT tillatt	ja	ja	ja
Ant variable modell/presolved	766/737	1532/1113	1532/1500
Ant restriksjoner modell/presolved	846/817	2077/1661	1690/1661
Løsning	optimal	lovlig	ingen løsning
Løsningstid	1476,1s	10653,4s	10000s
LP-relaksert løsning	4,41	8,82	8,82
Ant vedlikehold	5	12	-
Ant Vedlikehold hos NSB	6	12	12
Ant TT-km	1109,4	5301,6	-

ikke prioritert nå siden hovedfokuset har vært på å lage en modell som tar hensyn til vedlikeholdsplanleggingen, og fordi vår kilometerberegning vil ha noen små avvik i forhold til virkeligheten.

For å få vedlikeholdsplanleggingen best mulig, hadde det vært ønskelig å løse alle togtypene i samme modell. Dette har vi ikke klart siden problemet er så stort. Togtypen 69-3 er i seg selv så stor at vi ikke finner en lovlig løsning, se tabell 6.10. For å løse hele dette vedlikeholdsproblemet må modellen vår utbedres. Er alternativ er å lage flere mindre datasett og løse disse sekvensielt hver for seg. Hvert nytt datasett må dermed tilpasses de foregående løsningene. Så lenge modellen kun klarer å løse små problem kan denne jobben bli relativt arbeidskrevende. Det er heller ikke sikkert at en slik løsningsstratgi vil finne en lovlig eller en god løsning. Dermed vil en del av gevinsten ved å innføre et beslutningsstøttesystem forsvinne.

En alternativ framgangsmåte er å løse modellen med en mer effektiv løsningsstrategi. Xpress bruker Branch and Bound som løsningsstrategi og som vi har sett er denne ikke effektiv nok. I kapittel 3 beskriver vi ulike løsningsstrategier og et mulig videre arbeid kan være å finne løsningsstrategier som passer til dette problemet.

6.4 Utvidelse av modell

Modellen vår kan utvides videre med at man kan velge hvilken togtype som skal kjøre ulike dagsløp. Dette betyr i praksis at vedlikeholdsplanleggingen blir integrert med resten

Tabell 6.8: Løsning for flertypeproblemet med inndatasett typer 6

	1	2	3
Modell	**Flere typer**	**Flere typer**	**Flere typer**
Ant uker	en uke	to uker	to uker
Datasett	typer6	typer6	typer6
DLnoder/VLHnoder	28/36	56/72	56/72
Like uker	-	ja	nei
TT tillatt	ja	ja	ja
Ant variable modell/presolved	333/292	666/540	666/621
Ant restriksjoner modell/presolved	440/383	925/765	877/814
Løsning	optimal	lovlig	lovlig
Løsningstid	1,3s	15421,3s	9130,3s
LP-relaksert løsning	2,12	4,25	4,25
Ant vedlikehold	3	6	6
Ant vedlikehold hos NSB	3	6	6
Ant TT-km	547,9	1254	1254,2

av materiellplanleggingen. Da må en i tillegg til de restriksjonene vi har tatt med, også ta hensyn til de resterende kravene som NSB har til materiellplanlegging. Dette kan være krav som etterspørsel fra kundene og toglengde.

Tabell 6.9: Løsning for flertypeproblemet med inndatasett typer 5

	1	2
Modell	**Flere typer**	**Flere typer**
Ant uker	en uke	en uke
Datasett	typer5	typer5
DLnoder/VLHnoder	42/42	42/42
Like uker	-	-
TT tillatt	ja	nei
Ant variable modell/presolved	756/664	704/608
Ant restriksjoner modell/presolved	909/812	857/744
Løsning	lovlig	lovlig
Løsningstid	5340,0s	5158,8s
LP-relaksert løsning	3,61	3,61
Ant vedlikehold	5	5
Ant vedlikehold hos NSB	5	5
Ant TT-km	437,6	425,2

Tabell 6.10: Løsning for problemet med store sett med inndata

Modell	**En type**	**En type**	**En type**	**Flere typer**
Ant uker	en uke	en uke	en uke	en uke
Datasett	typer-692	typer-72	typer-693[1]	typerALLE
DLnoder/VLHnoder	95/64	105/42	326/64	526/64
Like uker	-	-	-	-
TT tillatt	ja	ja	ja	ja
Ant variable modell/presolved	4090/3796	3131/2928	23201/22301	30436/29039
Ant restriksjoner modell/presolved	4252/2531	3286/3102	23475/18516	30919/23317
Løsning	optimal	lovlig	ingen løsning	ingen løsning
Løsningstid	155,8s	73861,1s	84789s	-
LP-relaksert løsning	1,65	7,35	19,20	28,19
Ant vedlikehold	2	9	-	-
Ant Vedlikehold hos NSB	2	9	16	28
Ant TT-km	0	955,2km	-	-

Modell med inndata typer-693 ble kjørt på en maskin med 6GB minne og to 3GHz Intel Xeon prosessorer. NSB har brukt kilometergrense 10.000 for type 69 i stedet for 6.000 som vi har brukt.

Kapittel 7

Konklusjon

Denne prosjektoppgaven er et første forsøk på å lage et verktøy som kan brukes som besluttningsstøtte for vedlikeholdsplanlegging i NSB.

Ut ifra de artiklene vi har lest har de fleste materiellplanleggingsproblemene blitt formulert ved hjelp av en nettverksformulering. Vi har i midlertidig ikke funnet noe godt eksempel på en modellformulering som dekker de samme kravene som vedlikeholdsproblemet til NSB. Det var dermed vanskelig å forutsi hvor vanskelig dette problemet er å løse.

I dette prosjektet har vi formulert og implementert en nettverksformulering for vedlikeholdsproblemet til NSB. Ved testing klarer modellen vår å løse små datasett til optimalitet. Den klarer også å finne lovlige løsninger for flere større problemer. Antall vedlikehold i disse løsningene er like mange eller færre enn det NSB planlegger å gjennomføre. Disse delløsningene viser at det er potensial for å redusere antallet vedlikehold ved hjelp av optimering. Modellen klarer ikke å løse hele vedlikeholdsproblemet til NSB. For å løse hele vedlikeholdsproblemet til NSB trengs det derfor en mer effektiv løsningsstrategi. En annen mulighet er å angripe problemet med en alternativ modellformulering. Derfor har vi valgt å gi et forslag til en modellformulering som et set partitioning-problem. Dennee modellen tar ikke hensyn til alle kravene til vedlikeholdsproblemet, og kan sees på som en start på videre arbeid.

Modellen vår kan brukes både på ferdige planer og i en testfase i planleggingen. Modellen vil gi svar på hvilke dagsløp som kan kobles sammen, hvilke vedlikehold som skal brukes på vedlikeholdsbasene, hvor mange vedlikehold som trengs for å kjøre dagsløpene og hvilke tomtog som da må kjøres.

Bibliografi

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

K. Aschehoug and S. Lauritzen. Kvantifiserte effekter vedrørende innfasing av Type 72 i lokaltrafikken på Østlandet, 2002.

T. E. Bartlett and A. Charnes. Cyclic scheduling and combinatorial topology: Assignment and routing og motive power to meet scheduling and maintenance requirements; ii Generalization and analysis. *Naval research logistics quarterly*, (4):207–220, 1957.

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.

C. M. Berg and Arne Løkketangen. Bruk av tabusøk og critical event memory på set partitioning-problemet. Internett. http://www.nik.no/2002/Berg.pdf.

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.

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

D. Huisman, R. Freling, and A. P. M. Wagelmans. Multi-depot integrated vehicle and crew scheduling. Report EI2003-02, Econometric Institute, 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. *Optimal Vehicle Scheduling in Public Transit.* Doktoravhandling, Technische Universität Berlin, 1998.

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

A. Nõu, J. Desrosiers, and F. Soumis. Weekly locomotive scheduling at swedish state railways. GERAD G-97-35, École des Hautes Ètudes Commerciales de Montrèal, 1997.

B. Nygreen. Stykkevis lineærisering av dekomponerbare programmeringsproblem. Meddelelse nr 25, Institutt for sosialøkonomi, NTH, 1973.

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

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

I.-A. F. Sætermo and A. Tomasgard. Nsb drift og teknikk - forslag til endringer i planprosessen. Rapport STF 38 F01616, SINTEF Teknologiledelse, 2001.

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

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

H. P. Williams. *Model Building in Mathematical Programming.* John Wiley & Sons Ltd, Chichester, 4th edition, 1999.

Vedlegg A: Ordliste

Balanse: Antall tog på stasjonen til enhver tid er slik at rutetilbudet kan kjøres.

Dagsløp: Sekvens av turer på en dag.

Deling: Etterspørselen varierer fra tur til tur, og noen turer krever flere togsett enn andre turer. Deling er å dele togsettene som har kjørt turer sammen.

Driftspausebasert vedlikehold: Dette er vedlikehold som blir utført i driftspauser i materiellturneringen. Togsettet blir ikke tatt ut fra en turnering ('fritatt' for kjøring) når det skal vedlikeholdes.

Grunnplan: Det kommer en ny grunnplan hver januar og juni. Grunnplanen er ruteplanen for en vanlig uke, uten helligdager.

Grunnuke: Lengden på materiellturneringen som skal gjentas i perioden.

Hovedbestilling: En bestilling av togruter som NSB oversender til Jernbaneverket.

Materiellplan: Plan som viser hvordan togmateriellet turnerer for hver togtype.

Personellplan: Plan som viser de ulike skiftene for personalet.

Reservebeholdning: Togsett, lokomotiv og vogner som i utgangspunktet ikke inngår i en turnering, men er reserve.

Ruteplan: Ruteplan er den informasjonen som publikum får, med oversikt over hver enkelt togtur. Den viser startstasjon, endestasjon, starttid, sluttid og stasjoner der toget stopper underveis med tilhørende tidspunkter.

Skjøting: Etterspørselen varierer fra tur til tur, og noen turer krever flere togsett enn andre turer. Skjøting er å sette sammen togsett når flere skal kjøre turer sammen.

Togsett: Er et motorvognsett og består av 2, 3 eller 4 vogner. Togsettene kan kjøres i begge retninger, siden begge endene inneholder styringsmekanisme. Et togsett kan ikke deles opp i mindre enheter. For lokaltrafikken på Østlandet er togsettene type 69-2, 69-3 og 72.

Tomtogskjøring: Forflytning av tog/togsett utenom ruteplan og uten passasjerer.

Tur: En togreise fra ruteplanen angitt med startsted og sluttsted, starttidspunkt og sluttidspunkt og stasjoner underveis.

Turnering: En materiellturnering er lik en materiellplan.

Vedlikeholdsbase: En stasjon/stoppested der vedlikehold kan bli foretatt.

Vedlegg B: Kode

```
model NSB
uses "mmxprs"
uses "mmsystem"

(!-----------------------------------------------------
Modell for å optimere vedlikehold

Laget av Janne Bjorvand og Marianne Risberg
Prosjektoppgave NTNU høsten 2004
-----------------------------------------------------!)


(!
For å få programmet til å kjøre, må det bestemmes et sett med parametre.
Det er ulike inndata man kan kjøre på, og det er forklart hva de andre
parametrene må settes til for å få kjørt modellen der man bestemmer
parametrene.
!)


parameters

!-------------------------------------------------------------
! Alle togene med Togtype 72
! Sett med 15 dagsløpsnoder hver dag, i tillegg til alle mulige
! vedlikehold på Sundland
! TOMTOG = 1
!-------------------------------------------------------------
!SETT1 = 'typer_72.dat'

(!-------------------------------------------
Alle togene med togtype 69-2
TOMTOG = 1
-------------------------------------------!)
!SETT1 = 'typer_692.dat'
```

```
(!-----------------------------------------
Alle togene med togtype 69-3
TOMTOG = 1
-----------------------------------------!)
!SETT1 = 'typer_693.dat'


(!-----------------------------------------------------------
  Togtype 72
  Sett med to 14 Dagsløp og 14 Vedlikehold,
  TOMTOG = 1
-----------------------------------------------------------!)
!SETT1 = 'typer1.dat'

(!-----------------------------------------
 Togtype 72
 Sett med 28 Dagsløp og 14 Vedlikehold
 TOMTOG = 1
----------------------------------------- !)
!SETT1 = 'typer2.dat'

(!-----------------------------------------
 Togtype 72
 Sett med 28 Dagsløp og 14 Vedlikehold
 med TOMTOG, skal ikke generere tomtog i denne løsningen
 TOMTOG=0 , kan også kjøres med TOMTOG = 1
-----------------------------------------!)
!SETT1 = 'typer3a.dat'

(!-----------------------------------------
 Togtype 72 (samme tog som i typer 3a)
 Sett med 28 Dagsløp og 14 Vedlikehold
 uten TOMTOG, skal ikke generere tomtog i denne løsningen
 TOMTOG=1
-----------------------------------------!)
!SETT1 = 'typer3c.dat'


(!-----------------------------------------
 Togtype 72 og 693
 Sett med 42 Dagsløp og 42 Vedlikehold
 TOMTOG = 0 , kan også kjøres med TOMTOG = 1
----------------------------------------- !)
!SETT1 = 'typer5.dat'
```

```
(!-----------------------------------------
 Togtype 72 og 693
 Sett med 28 Dagsløp og 36 Vedlikehold
 TOMTOG = 1
------------------------------------------ !)
!SETT1 = 'typer6.dat'

(!-----------------------------------------
Togtype 72
Sett med 42 dagsløp og 14 vedlikehold
Tomtog = 1
-----------------------------------------------!)
!SETT1 = 'typer7.dat'

(!-----------------------------------------
Togtype 72
Sett med 56 dagsløp og 14 vedlikehold
Tomtog = 1
-----------------------------------------------!)
!SETT1 = 'typer8.dat'

(!-----------------------------------------
Togtype 72 - 693 -692
Sett med alle dagsløpene og vedlikeholdene for lokaltog i Oslo
Tomtog = 1
-----------------------------------------------!)
!SETT1 = 'typer_alle.dat'


!-----------------------------------------------
! Bestemme om det skal kjøre for en eller to uker
! PTOTAL kan være enten 7 eller 14
!-----------------------------------------------
PTOTAL = 14
TTOTAL = 7

!-------------------------------------------------------------------
! hvis LIKEUKER er ulik en, så gjøres ikke noe for å få like uker
!   --> Går ikke inn i restriksjonen som sørger for dette
!-------------------------------------------------------------------

LIKEUKER = 1

!-------------------------------------------------------------------
```

```
! hvis TOMTOG er ulik NULL, så vil tomtog inkluderes i modellen
!    --> tid mellom to stasjoner settes veldig høyt slik at de ikke
! blir lovlige innoder
!-------------------------------------------------------------------

TOMTOG = 0

end-parameters

declarations



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

HKAP: array(STASJONER) of integer  !hensettingskapasitet

MATKAP: array(TYPE) of integer  !materiellltilgjengelighet

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



!Beslutningsvariable:

x:  array(NODER,NODER,TYPE) of mpvar !kobling mellom to noder, flyt
```

```
z: array(NODER,TYPE) of mpvar !teller kilometer

end-declarations

initializations from SETT1
DAG
TYPE
STASJONER
NODER
DAGSLOPSNODER
VLHNODER
KMGRENSE
STARTTID
SLUTTID
STARTSTED
SLUTTSTED
DAGSLOPSDAG
NODETYPE
TID
KM
LENGDE
HKAP

end-initializations



!-------------------------------------------------------
!genererer noen konstanter som vi trenger i beregningene
!-------------------------------------------------------

ANTALLDAGSLOP := getsize(DAGSLOPSNODER)
ANTALLVLHNODER := getsize(VLHNODER)
ANTALLNODER := getsize(NODER)

!-----------------------------------------
!lager lengde hvis PTOTAL>TTOTAL
!dvs genererer lengden for nodene i uke 2
!-----------------------------------------
forall(j in NODER|j<=ANTALLDAGSLOP and j>0)do

if (PTOTAL>TTOTAL) then

NYj := j + ANTALLNODER
LENGDE(NYj) :=LENGDE(j)
```

```
end-if
end-do

!-----------------------------------------
!lager startsted og sluttsted hvis PTOTAL>TTOTAL
!(for nodene i uke 2)
!-----------------------------------------
forall(j in NODER|j <=ANTALLNODER) do

if(PTOTAL>TTOTAL)then

NYj := j+ ANTALLNODER
STARTSTED(NYj) := STARTSTED(j)
SLUTTSTED(NYj) := SLUTTSTED(j)
STARTTID(NYj) := STARTTID(j)
SLUTTID(NYj) := SLUTTID(j)

end-if

end-do

!---------------------------
!Generere TIDTT
!---------------------------
forall(j in NODER|j<=ANTALLNODER, i in NODER|i<>j and i<=ANTALLNODER) do

 !for å begrense tabellen litt :-)
 if(DAGSLOPSDAG(j)>=DAGSLOPSDAG(i) or
  DAGSLOPSDAG(i) = TTOTAL and DAGSLOPSDAG(j)=1)then

if(STARTSTED(j)= SLUTTSTED(i)) then
TIDTT(i,j) := 0

else
c := STARTSTED(j)
d := SLUTTSTED(i)

!krever at ingen dagsløp starter eller slutter
!på vedlikeholdsbasen
if(TOMTOG=0) then
if((c = "Filipstad") or (d= "Filipstad") or
(d= "Sundland")or (c= "Sundland")) then
TIDTT(i,j) := TID(d,c)
else TIDTT(i,j) :=50
```

```
end-if

else
TIDTT(i,j) := TID(d,c)
end-if
end-if
  end-if
end-do


!-------------------------------------------------
!innoder for alle dagsløpsnoder
!-------------------------------------------------

forall(j in NODER|j<=ANTALLDAGSLOP and j>0, k in TYPE) do
KOLONNE2 :=1

forall(i in NODER| i<>j and i<=ANTALLNODER) do

!må være samme type
if(NODETYPE(i,k)=NODETYPE(j,k) and NODETYPE(j,k)=1)then

U(i):= DAGSLOPSDAG(i)
V(j):= DAGSLOPSDAG(j)


!Innoder fra dagen før.
if(LENGDE(j)>0) then !1

if (V(j)>U(i) and U(i)<>TTOTAL) then !3

   !hvis det er tomt dagsløp dagen før, vil dette være innode
   if (V(j)=U(i)+1)  then
     if(i <= ANTALLDAGSLOP and LENGDE(i)=0) then
   DINNODER(j,KOLONNE2,k):=(i)
ADINNODER(j,k):=KOLONNE2

if(PTOTAL>TTOTAL)then
NYj:=j+ANTALLNODER
NYi:=i+ANTALLNODER
DINNODER(NYj,KOLONNE2,k):=(NYi)
ADINNODER(NYj,k):=KOLONNE2
  end-if
KOLONNE2 +=1
```

```
   !lengden av i er større enn null eller
   !i er vedlikehold
   elif(i <=ANTALLNODER) then

   if (STARTTID(i)>SLUTTID(i))then
   SLUTTID1(i) := SLUTTID(i) + 24

   else SLUTTID1(i) := SLUTTID(i)
   end-if

   STARTTID1(j) := STARTTID(j) +24

   if(STARTTID1(j) >= SLUTTID1(i) + TIDTT(i,j))then
DINNODER(j,KOLONNE2,k):=(i)
ADINNODER(j,k):=KOLONNE2

if(PTOTAL>TTOTAL)then
NYj:=j+ANTALLNODER
NYi:=i+ANTALLNODER
DINNODER(NYj,KOLONNE2,k):=(NYi)
ADINNODER(NYj,k):=KOLONNE2
  end-if
KOLONNE2 +=1
   end-if
end-if
   end-if



!Innoder mellom søndag og mandag hvis PTOTAL=TTOTAL

elif (TTOTAL=PTOTAL and V(j)=1 and U(i)=TTOTAL) then
if(i<=ANTALLDAGSLOP and LENGDE(i)=0) then
DINNODER(j,KOLONNE2,k):=(i)
ADINNODER(j,k):=KOLONNE2
KOLONNE2+=1

elif(i <=ANTALLNODER)then
if (STARTTID(i)>SLUTTID(i))then
   SLUTTID1(i) := SLUTTID(i) + 24
   else SLUTTID1(i) := SLUTTID(i)
   end-if

  STARTTID1(j) := STARTTID(j) +24
```

```
if (STARTTID1(j)>=SLUTTID1(i)+TIDTT(i,j)) then
DINNODER(j,KOLONNE2,k):=(i)
ADINNODER(j,k):=KOLONNE2
KOLONNE2+=1
end-if
end-if


!innoder mellom søndag og mandag hvis det er flere uker

elif(PTOTAL>TTOTAL and V(j)=1 and U(i)=TTOTAL)then

if(i<=ANTALLDAGSLOP and LENGDE(i) = 0) then

NYj := j+ ANTALLNODER
NYi := i + ANTALLNODER
DINNODER(j,KOLONNE2,k):=(NYi)
DINNODER(NYj,KOLONNE2,k):=(i)
ADINNODER(j,k):=KOLONNE2
ADINNODER(NYj,k):=KOLONNE2
KOLONNE2 +=1

elif(i<=ANTALLNODER)then

if (STARTTID(i)>SLUTTID(i))then
   SLUTTID1(i) := SLUTTID(i) + 24

   else SLUTTID1(i) := SLUTTID(i)
   end-if

STARTTID1(j) := STARTTID(j) + 24

if(STARTTID1(j)>=SLUTTID1(i)+TIDTT(i,j)) then
NYj := j+ ANTALLNODER
NYi := i + ANTALLNODER
DINNODER(j,KOLONNE2,k):=(NYi)
DINNODER(NYj,KOLONNE2,k):=(i)
ADINNODER(j,k):=KOLONNE2
ADINNODER(NYj,k):=KOLONNE2
KOLONNE2 +=1
end-if
end-if

!Innoder som starter samme dag
```

```
!Setter tomtogskjøring mellom nodene som starter sammen dag

!Man kan ikke koble sammen et tomt dagsløp
!med et annet dagsløp samme dag
elif(U(i)=V(j)) then
!Sjekker om det er tilstrekkelig med tid for å koble
!sammen to dagsløp på samme dag


!-------------------------------------------------------------
!Når dagslopet har tom lengde, er STARTTID(i)=SLUTTID(i)
!-------------------------------------------------------------

if (STARTTID(j)>=SLUTTID(i)+TIDTT(i,j) and
SLUTTID(i)>STARTTID(i)) then

DINNODER(j,KOLONNE2,k):=(i)
ADINNODER(j,k):=KOLONNE2

if(PTOTAL>TTOTAL) then
NYj:=j+ANTALLNODER
NYi:=i+ANTALLNODER
  DINNODER(NYj,KOLONNE2,k):=(NYi)
ADINNODER(NYj,k):=KOLONNE2
end-if
KOLONNE2+=1
  end-if

end-if

!for tomme dagsløp kan det være flere innoder...
elif(LENGDE(j)=0) then
if (V(j)>U(i) and U(i)<>TTOTAL) then

if (V(j)=U(i)+1)  then
DINNODER(j,KOLONNE2,k):=(i)
ADINNODER(j,k):=KOLONNE2

if(PTOTAL>TTOTAL)then

NYj:=j+ANTALLNODER
NYi:=i+ANTALLNODER
DINNODER(NYj,KOLONNE2,k):=(NYi)
ADINNODER(NYj,k):=KOLONNE2
```

```
end-if
KOLONNE2 +=1


end-if

!Innoder mellom søndag og mandag hvis PTOTAL=TTOTAL
elif (TTOTAL=PTOTAL and V(j)=1 and U(i)=TTOTAL) then
DINNODER(j,KOLONNE2,k):=(i)
ADINNODER(j,k):=KOLONNE2 !her var det en feil
KOLONNE2+=1

elif(PTOTAL>TTOTAL and V(j)=1 and U(i)=TTOTAL)then

NYj := j+ ANTALLNODER
NYi := i + ANTALLNODER
DINNODER(j,KOLONNE2,k):=(NYi)
DINNODER(NYj,KOLONNE2,k):=(i)
ADINNODER(j,k):=KOLONNE2
ADINNODER(NYj,k):=KOLONNE2
KOLONNE2 +=1


!Innoder som starter samme dag
!Setter tomtogskjøring mellom nodene som starter sammen dag

!Hvis dagslopsdag for i og j er det samme
elif(U(i)=V(j)) then


!legge til alle VLHNODER samme dag som lovlige inputnoder
if(i>ANTALLDAGSLOP and i<=ANTALLNODER) then
DINNODER(j,KOLONNE2,k):=(i)
ADINNODER(j,k):=KOLONNE2
if(PTOTAL>TTOTAL) then
NYj:=j+ANTALLNODER
NYi:=i+ANTALLNODER
DINNODER(NYj,KOLONNE2,k):=(NYi)
ADINNODER(NYj,k):=KOLONNE2
end-if
KOLONNE2+=1
  end-if

end-if
end-if
```

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



!-------------------------------------------
!LAGER INNODER FOR ALLE VEDLIKEHOLDSNODENE
!-------------------------------------------

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

 KOLONNE2 :=1

!Kan ikke koble et vedlikehold til et vedlikehold.

forall(i in NODER|i<=ANTALLDAGSLOP) do

!må være samme type
if(NODETYPE(i,k)=NODETYPE(j,k) and NODETYPE(j,k)=1)then

U(i):= DAGSLOPSDAG(i)
V(j):= DAGSLOPSDAG(j)

!Innoder fra dagen før.

if(LENGDE(i)>0) then
if (V(j)>U(i) and U(i)<>TTOTAL) then
   if (V(j)=U(i)+1)  then

   if(STARTTID(i)>SLUTTID(i)) then
   SLUTTID1(i) := SLUTTID(i) + 24
   else SLUTTID1(i) := SLUTTID(i)
   end-if
   STARTTID1(j) := STARTTID(j) + 24

   if(STARTTID1(j) >= SLUTTID1(i) + TIDTT(i,j)) then
VINNODER(j,KOLONNE2,k):=(i)
AVINNODER(j,k):=KOLONNE2
if(PTOTAL>TTOTAL)then
NYj:=j+ANTALLNODER
NYi:=i+ANTALLNODER
VINNODER(NYj,KOLONNE2,k):=(NYi)
AVINNODER(NYj,k):=KOLONNE2
  end-if
```

```
KOLONNE2 +=1
end-if
end-if
!Innoder mellom søndag og mandag

   elif (TTOTAL=PTOTAL and V(j)=1 and U(i)=TTOTAL) then

if(STARTTID(i)>SLUTTID(i)) then
   SLUTTID1(i) := SLUTTID(i) + 24
   else SLUTTID1(i) := SLUTTID(i)
   end-if
STARTTID1(j) := STARTTID(j) + 24

   if(STARTTID1(j) >= SLUTTID1(i) + TIDTT(i,j)) then

VINNODER(j,KOLONNE2,k):=(i)
AVINNODER(j,k):=KOLONNE2
KOLONNE2+=1
end-if
elif(PTOTAL>TTOTAL and V(j)=1 and U(i)=TTOTAL)then

  if(STARTTID(i)>SLUTTID(i)) then
   SLUTTID1(i) := SLUTTID(i) + 24
   else SLUTTID1(i) := SLUTTID(i)
   end-if
   STARTTID1(j) := STARTTID(j) + 24
   if(STARTTID1(j) >= SLUTTID1(i) + TIDTT(i,j)) then
VINNODER(j,KOLONNE2,k):=(i+ANTALLNODER)
VINNODER(j+ANTALLNODER,KOLONNE2,k):=(i)
AVINNODER(j,k):=KOLONNE2
AVINNODER(j+ANTALLNODER,k):=KOLONNE2
KOLONNE2 +=1
end-if

!Innoder som starter sammen dag
!Setter tomtogskjøring mellom nodene som starter sammen dag

!Hvis dagslopsdag for i og j er det samme
elif(U(i)=V(j)) then

!Sjekker om det er tilstrekkelig med tid for å koble
! sammen to dagsløp på samme dag
if (STARTTID(j)>=SLUTTID(i)+TIDTT(i,j) and
SLUTTID(i)>STARTTID(i)) then
```

```
VINNODER(j,KOLONNE2,k):=(i)
AVINNODER(j,k):=KOLONNE2
if(PTOTAL>TTOTAL) then
NYj:=j+ANTALLNODER
NYi:=i+ANTALLNODER
VINNODER(NYj,KOLONNE2,k):=(NYi)
AVINNODER(NYj,k):=KOLONNE2
  end-if
KOLONNE2+=1
  end-if

end-if



elif(LENGDE(i)=0) then

if (V(j)>U(i) and U(i)<>TTOTAL) then
if (V(j)=U(i)+1)  then

VINNODER(j,KOLONNE2,k):=(i)
AVINNODER(j,k):=KOLONNE2
if(PTOTAL>TTOTAL)then
NYj:=j+ANTALLNODER
NYi:=i+ANTALLNODER
VINNODER(NYj,KOLONNE2,k):=(NYi)
AVINNODER(NYj,k):=KOLONNE2
end-if
KOLONNE2 +=1

end-if

!Innoder mellom søndag og mandag

elif (TTOTAL=PTOTAL and V(j)=1 and U(i)=TTOTAL) then

VINNODER(j,KOLONNE2,k):=(i)
AVINNODER(j,k):=KOLONNE2
KOLONNE2+=1


elif(PTOTAL>TTOTAL and V(j)=1 and U(i)=TTOTAL)then
VINNODER(j,KOLONNE2,k):=(i+ANTALLNODER)
VINNODER(j+ANTALLNODER,KOLONNE2,k):=(i)
AVINNODER(j,k):=KOLONNE2
```

```
AVINNODER(j+ANTALLNODER,k):=KOLONNE2
KOLONNE2 +=1


!Innoder som starter samme dag
!Setter tomtogskjøring mellom nodene som starter samme dag

!Hvis dagslopsdag for i og j er det samme
elif(U(i)=V(j)) then

VINNODER(j,KOLONNE2,k):=(i)
AVINNODER(j,k):=KOLONNE2
if(PTOTAL>TTOTAL) then
NYj:=j+ANTALLNODER
NYi:=i+ANTALLNODER
VINNODER(NYj,KOLONNE2,k):=(NYi)
AVINNODER(NYj,k):=KOLONNE2
end-if
KOLONNE2+=1

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


!-----------------------------------------
!Lager utnoder for alle DAGSLOPnodene D-->N
!-----------------------------------------


forall(j in NODER|j<=ANTALLDAGSLOP, k in TYPE) do
KOLONNE:=1
forall(i in NODER| i<>j and i<=ANTALLNODER) do

!må være samme type
if(NODETYPE(i,k)=NODETYPE(j,k) and NODETYPE(j,k)=1)then

T(j):=DAGSLOPSDAG(j)
S(i):=DAGSLOPSDAG(i)

if(LENGDE(j)>0) then

!Kobler mellom to etterfølgende dager
```

```
if (S(i)>T(j) and T(j)<>TTOTAL) then
if (T(j)=S(i)-1) then
if(i<=ANTALLDAGSLOP and LENGDE(i)= 0) then
DUTNODER(j,KOLONNE,k):=(i)
ADUTNODER(j,k):=KOLONNE
if(PTOTAL>TTOTAL)then
NYj:=j+ANTALLNODER
NYi:=i+ANTALLNODER
DUTNODER(NYj,KOLONNE,k):=(NYi)
ADUTNODER(NYj,k):=KOLONNE
  end-if
KOLONNE +=1
!lengden på dagsløp er større enn null
elif(i <=ANTALLNODER)then
if (STARTTID(j)>SLUTTID(j)) then
   SLUTTID1(j) := SLUTTID(j)+24
   else SLUTTID1(j) := SLUTTID(j)
   end-if
   STARTTID1(i) := STARTTID(i) + 24

   if(STARTTID1(i)>=SLUTTID1(j) + TIDTT(j,i)) then
DUTNODER(j,KOLONNE,k):=(i)
ADUTNODER(j,k):=KOLONNE
if(PTOTAL>TTOTAL)then
NYj:=j+ANTALLNODER
NYi:=i+ANTALLNODER
DUTNODER(NYj,KOLONNE,k):=(NYi)
ADUTNODER(NYj,k):=KOLONNE
end-if
KOLONNE +=1
end-if
end-if
end-if

!Kobler mellom søndag og mandag
elif (TTOTAL=PTOTAL and T(j)=TTOTAL and S(i)=1) then
if(i<=ANTALLDAGSLOP and LENGDE(i) = 0) then
DUTNODER(j,KOLONNE,k):=(i)
ADUTNODER(j,k):=KOLONNE
KOLONNE +=1

elif(i<=ANTALLNODER) then
if (STARTTID(j)>SLUTTID(j)) then
   SLUTTID1(j) := SLUTTID(j)+24
   else SLUTTID1(j) := SLUTTID(j)
```

```
   end-if
   STARTTID1(i) := STARTTID(i) + 24

   if(STARTTID1(i)>=SLUTTID1(j) + TIDTT(j,i)) then

DUTNODER(j,KOLONNE,k):=(i)
ADUTNODER(j,k):=KOLONNE
KOLONNE +=1
end-if
end-if

elif(PTOTAL>TTOTAL and T(j)=TTOTAL and S(i)=1) then

if(i<=ANTALLDAGSLOP and LENGDE(i)=0) then

NYi:=i+ANTALLNODER
DUTNODER(j+ANTALLNODER,KOLONNE,k):=(i)
DUTNODER(j, KOLONNE,k):= (NYi)
ADUTNODER(j,k):=KOLONNE
ADUTNODER(j+ANTALLNODER,k) := KOLONNE
KOLONNE +=1


elif(i<=ANTALLNODER)then
if (STARTTID(j)>SLUTTID(j)) then
   SLUTTID1(j) := SLUTTID(j)+24
   else SLUTTID1(j) := SLUTTID(j)
   end-if
   STARTTID1(i) := STARTTID(i) + 24

   if(STARTTID1(i)>=SLUTTID1(j) + TIDTT(j,i)) then

NYi:=i+ANTALLNODER
DUTNODER(j+ANTALLNODER,KOLONNE,k):=(i)
DUTNODER(j,KOLONNE,k):= (NYi)
ADUTNODER(j,k):=KOLONNE
ADUTNODER(j+ANTALLNODER,k) := KOLONNE
KOLONNE +=1
end-if
end-if
!Kobler to samme dag
elif(T(j)=S(i)) then
!-------------------------------------------------------
!Når SLUTTID(j) = STARTTID(j), har dagsløpet lengden null
!-------------------------------------------------------
```

```
if(STARTTID(i)>=SLUTTID(j)+ TIDTT(j,i) and
SLUTTID(j)>STARTTID(j)) then

DUTNODER(j,KOLONNE,k):=(i)
ADUTNODER(j,k):=KOLONNE
if(PTOTAL>TTOTAL) then
NYj:=j+ANTALLNODER
NYi:=i+ANTALLNODER
DUTNODER(NYj,KOLONNE,k):=(NYi)
ADUTNODER(NYj,k):=KOLONNE
  end-if
KOLONNE +=1
end-if

end-if

elif(LENGDE(j)=0) then
!Kobler mellom to etterfølgende dager
if (S(i)>T(j) and T(j)<>TTOTAL) then
if (T(j)=S(i)-1) then

DUTNODER(j,KOLONNE,k):=(i)
ADUTNODER(j,k):=KOLONNE
if(PTOTAL>TTOTAL)then
NYj:=j+ANTALLNODER
NYi:=i+ANTALLNODER
DUTNODER(NYj,KOLONNE,k):=(NYi)
ADUTNODER(NYj,k):=KOLONNE
end-if
KOLONNE +=1
end-if

!Kobler mellom søndag og mandag
elif (TTOTAL=PTOTAL and T(j)=TTOTAL and S(i)=1) then

DUTNODER(j,KOLONNE,k):=(i)
ADUTNODER(j,k):=KOLONNE
KOLONNE +=1

elif(PTOTAL>TTOTAL and T(j)=TTOTAL and S(i)=1) then

NYi:=i+ANTALLNODER
DUTNODER(j+ANTALLNODER,KOLONNE,k):=(i)
DUTNODER(j, KOLONNE,k):= (NYi)
```

```
ADUTNODER(j,k):=KOLONNE
ADUTNODER(j+ANTALLNODER,k) := KOLONNE
KOLONNE +=1

!Kobler to samme dag
elif(T(j)=S(i)) then

if(i>ANTALLDAGSLOP and i<=ANTALLNODER)then
DUTNODER(j,KOLONNE,k):=(i)
ADUTNODER(j,k):=KOLONNE
if(PTOTAL>TTOTAL) then
NYj:=j+ANTALLNODER
NYi:=i+ANTALLNODER
DUTNODER(NYj,KOLONNE,k):=(NYi)
ADUTNODER(NYj,k):=KOLONNE
end-if
KOLONNE +=1
end-if

end-if

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



!-------------------------------------------------------
!UTNODER FOR ALLE VEDLIKEHOLDSNODENE
!-------------------------------------------------------

forall(j in NODER|j>ANTALLDAGSLOP and j<=ANTALLNODER, k in TYPE) do
KOLONNE:=1
forall(i in NODER|i<=ANTALLDAGSLOP) do !dagslopsnoder

!må være samme type
if(NODETYPE(i,k)=NODETYPE(j,k) and NODETYPE(j,k)=1)then

T(j):=DAGSLOPSDAG(j)
S(i):=DAGSLOPSDAG(i)

if (LENGDE(i)>0) then

!Kobler mellom to etterfølgende dager
```

```
if (S(i)>T(j)and T(j)<>TTOTAL) then
if (T(j)=S(i)-1) then

if (STARTTID(j)>SLUTTID(j)) then
   SLUTTID1(j) := SLUTTID(j)+24
   else SLUTTID1(j) := SLUTTID(j)
   end-if
   STARTTID1(i) := STARTTID(i) + 24

   if(STARTTID1(i)>=SLUTTID1(j) + TIDTT(j,i)) then

VUTNODER(j,KOLONNE,k):=(i)
AVUTNODER(j,k):=KOLONNE
if(PTOTAL>TTOTAL)then
NYj:=j+ANTALLNODER
NYi:=i+ANTALLNODER
VUTNODER(NYj,KOLONNE,k):=(NYi)
AVUTNODER(NYj,k):=KOLONNE
  end-if
KOLONNE +=1

end-if
end-if

!Kobler mellom søndag og mandag
elif (TTOTAL=PTOTAL and T(j)=TTOTAL and S(i)=1) then

if (STARTTID(j)>SLUTTID(j)) then
   SLUTTID1(j) := SLUTTID(j)+24
   else SLUTTID1(j) := SLUTTID(j)
   end-if
   STARTTID1(i) := STARTTID(i) + 24

   if(STARTTID1(i)>=SLUTTID1(j) + TIDTT(j,i)) then
VUTNODER(j,KOLONNE,k):=(i)
AVUTNODER(j,k):=KOLONNE
KOLONNE +=1
end-if

elif(PTOTAL>TTOTAL and T(j) = TTOTAL and S(i)= 1) then

if (STARTTID(j)>SLUTTID(j)) then
   SLUTTID1(j) := SLUTTID(j)+24
   else SLUTTID1(j) := SLUTTID(j)
   end-if
```

```
   STARTTID1(i) := STARTTID(i) + 24

   if(STARTTID1(i)>=SLUTTID1(j) + TIDTT(j,i)) then
NYi:=i+ANTALLNODER
VUTNODER(j,KOLONNE,k):=(NYi)
AVUTNODER(j,k):=KOLONNE
VUTNODER(j+ANTALLNODER,KOLONNE,k):=(i)
  AVUTNODER(j+ANTALLNODER,k):=KOLONNE

KOLONNE +=1

end-if


!Kobler to samme dag
elif(T(j)=S(i)) then


if(STARTTID(i)>=SLUTTID(j)+ TIDTT(j,i) and
SLUTTID(j)>STARTTID(j)) then

VUTNODER(j,KOLONNE,k):=(i)
AVUTNODER(j,k):=KOLONNE
if(PTOTAL>TTOTAL) then
NYj:=j+ANTALLNODER
NYi:=i+ANTALLNODER
VUTNODER(NYj,KOLONNE,k):=(NYi)
AVUTNODER(NYj,k):=KOLONNE
  end-if
KOLONNE +=1
end-if

end-if

elif (LENGDE(i)=0) then
if(T(j)<>TTOTAL)then

!Kobler mellom to etterfølgende dager
if (S(i)>T(j)) then
if (T(j)=S(i)-1) then

VUTNODER(j,KOLONNE,k):=(i)
AVUTNODER(j,k):=KOLONNE
if(PTOTAL>TTOTAL)then
NYj:=j+ANTALLNODER
```

```
NYi:=i+ANTALLNODER
VUTNODER(NYj,KOLONNE,k):=(NYi)
AVUTNODER(NYj,k):=KOLONNE
  end-if
KOLONNE +=1
end-if
end-if

!Kobler mellom søndag og mandag
elif (TTOTAL=PTOTAL and T(j)=TTOTAL and S(i)=1) then

VUTNODER(j,KOLONNE,k):=(i)
AVUTNODER(j,k):=KOLONNE
KOLONNE +=1

elif(PTOTAL>TTOTAL and T(j) = TTOTAL and S(i)= 1) then

NYi:=i+ANTALLNODER
VUTNODER(j,KOLONNE,k):=(NYi)
AVUTNODER(j,k):=KOLONNE
VUTNODER(j+ANTALLNODER,KOLONNE,k):=(i)
AVUTNODER(j+ANTALLNODER,k):=KOLONNE
KOLONNE +=1

!Kobler to samme dag
elif(T(j)=S(i)) then
!alle dagsløpene som har lengde null er lovlige
!utnoder fra vedlikehold

VUTNODER(j,KOLONNE,k):=(i)
AVUTNODER(j,k):=KOLONNE
if(PTOTAL>TTOTAL) then
NYj:=j+ANTALLNODER
NYi:=i+ANTALLNODER
VUTNODER(NYj,KOLONNE,k):=(NYi)
AVUTNODER(NYj,k):=KOLONNE
end-if
KOLONNE +=1

end-if

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

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

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

create(x(DINNODER(j,a,k),j,k))

if (PTOTAL>TTOTAL) then
v:= j + ANTALLNODER
create(x(DINNODER(v,a,k),v,k))
end-if
end-do
end-do


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

if (PTOTAL>TTOTAL) then
v:= j+ ANTALLNODER
create(x(VINNODER(v,a,k),v,k))
end-if
end-do
end-do


forall(j in NODER|j<=ANTALLDAGSLOP, k in TYPE) do
forall(a in 1..ADINNODER(j,k))do
x(DINNODER(j,a,k),j,k)is_binary

if(PTOTAL>TTOTAL) then
v:= j + ANTALLNODER
x(DINNODER(v,a,k),v,k)is_binary
end-if
end-do
end-do

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

```
x(VINNODER(j,a,k),j,k) is_binary

if(PTOTAL>TTOTAL) then
v:= j+ ANTALLNODER
x(VINNODER(v,a,k),v,k) is_binary
end-if
end-do


forall(i in NODER|i <=ANTALLDAGSLOP,k in TYPE) do
create(z(i,k))
if (PTOTAL>TTOTAL) then
v:= i + ANTALLNODER
   create(z(v,k))
end-if
end-do


forall(i in NODER|i<=ANTALLDAGSLOP, k in TYPE) do
z(i,k)<= KMGRENSE(k)
if (PTOTAL>TTOTAL) then
v:= i + ANTALLNODER
z(v,k)<= KMGRENSE(k)
end-if
end-do


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

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

TOTALFLYTDAGSLOP(j,k):=
SUM(a in 1..ADINNODER(j,k)) x(DINNODER(j,a,k),j,k) -
SUM(b in 1..ADUTNODER(j,k)) x(j,DUTNODER(j,b,k),k)=0

if (PTOTAL>TTOTAL) then
v:=j+ANTALLNODER
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-if
end-do
```

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

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

if (PTOTAL>TTOTAL) then
v:=j+ANTALLNODER
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-if
end-do

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

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

if (PTOTAL>TTOTAL) then
NYj:=j+ANTALLNODER

TOTALINNFLYTDAGSLOP(NYj,k):=
 SUM(a in 1..ADINNODER(NYj,k)) x(DINNODER(NYj,a,k),NYj,k)=1
end-if
end-do

forall(j in NODER|j>ANTALLDAGSLOP and j<=ANTALLNODER) do

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

if(PTOTAL>TTOTAL) then
NYj:=j+ANTALLNODER

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



!-------------
```

```
!lager KMTT
!-------------
forall(j in NODER|j<=ANTALLDAGSLOP,k in TYPE, a in 1..ADINNODER(j,k))do

if(SLUTTSTED(DINNODER(j,a,k)) = STARTSTED(j)) then

KMTT(DINNODER(j,a,k),j) := 0

if(PTOTAL>TTOTAL)then
v:= j + ANTALLNODER
KMTT(DINNODER(v,a,k),v):=0
end-if

else
if(PTOTAL=TTOTAL) then
p := SLUTTSTED(DINNODER(j,a,k))
q := STARTSTED(j)
KMTT(DINNODER(j,a,k),j) := KM(p,q)

else !(PTOTAL>TTOTAL)then

p := SLUTTSTED(DINNODER(j,a,k))
q := STARTSTED(j)
v:= j + ANTALLNODER
KMTT(DINNODER(j,a,k),j) := KM(p,q)
KMTT(DINNODER(v,a,k),v) := KM(p,q)
end-if

end-if
end-do

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

if(SLUTTSTED(VINNODER(j,a,k)) = STARTSTED(j)) then

KMTT(VINNODER(j,a,k),j) := 0

if(PTOTAL>TTOTAL)then
v:= j + ANTALLNODER
KMTT(VINNODER(v,a,k),v):=0
end-if
else
if(PTOTAL=TTOTAL) then
p := SLUTTSTED(VINNODER(j,a,k))
```

```
q := STARTSTED(j)
KMTT(VINNODER(j,a,k),j) := KM(p,q)

else

p := SLUTTSTED(VINNODER(j,a,k))
q := STARTSTED(j)
v:= j + ANTALLNODER
KMTT(VINNODER(j,a,k),j) := KM(p,q)
KMTT(VINNODER(v,a,k),v) := KM(p,q)
end-if

end-if
end-do


!----------------------------------------
!ulineære restriksjoner og BIGM-variabler
!----------------------------------------
forall(j in NODER|j<=ANTALLDAGSLOP,k in TYPE, a in 1..ADINNODER(j,k)) do

M1(DINNODER(j,a,k),j,k):=
KMGRENSE(k) + KMTT(DINNODER(j,a,k),j) + LENGDE(j)

if(PTOTAL>TTOTAL) then
NYj:=j+ANTALLNODER
M1(DINNODER(NYj,a,k),NYj,k):=
KMGRENSE(k) + KMTT(DINNODER(NYj,a,k),NYj) + LENGDE(j)

end-if
end-do

forall(i in NODER|i>ANTALLDAGSLOP and
   i<=ANTALLDAGSLOP+ANTALLVLHNODER,k in TYPE, a in 1..AVUTNODER(i,k)) do

M2(i,VUTNODER(i,a,k)):=
KMTT(i,VUTNODER(i,a,k))+ LENGDE(VUTNODER(i,a,k))

if(PTOTAL>TTOTAL) then
NYi:=i+ANTALLNODER
M2(NYi,VUTNODER(NYi,a,k)):=
KMTT(NYi,VUTNODER(NYi,a,k))+LENGDE(VUTNODER(NYi,a,k))
end-if

end-do
```

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

if(PTOTAL=TTOTAL)then
M3(VINNODER(j,a,k),j):=KMTT(VINNODER(j,a,k),j)

elif (PTOTAL>TTOTAL) then
v:=j+ANTALLNODER
M3(VINNODER(j,a,k),j):=KMTT(VINNODER(j,a,k),j)
M3(VINNODER(v,a,k),v):=KMTT(VINNODER(v,a,k),v)
end-if
end-do


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

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

if(DINNODER(j,a,k)<=ANTALLDAGSLOP or(DINNODER(j,a,k)>ANTALLNODER and
DINNODER(j,a,k) <= ANTALLNODER+ANTALLDAGSLOP)) then

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


if(PTOTAL>TTOTAL) then
v:=j+ANTALLNODER
Tellerkm(DINNODER(v,a,k),v,k):=
  z(DINNODER(v,a,k),k)-z(v,k)+
  M1(DINNODER(v,a,k),v,k) * x(DINNODER(v,a,k),v,k)<=
  M1(DINNODER(v,a,k),v,k) - KMTT(DINNODER(v,a,k),v)-LENGDE(j)

end-if
end-if
end-do

!-----------------------------------
! Fra vedlikehold til dagsløp
```

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

VDTellerkm(j,VUTNODER(j,a,k),k):=
-z(VUTNODER(j,a,k),k)+
M2(j,VUTNODER(j,a,k)) * x(j, VUTNODER(j,a,k),k)<=
M2(j,VUTNODER(j,a,k)) - KMTT(j,VUTNODER(j,a,k)) -
LENGDE(VUTNODER(j,a,k))


if(PTOTAL>TTOTAL) then
v:=j+ANTALLNODER
VDTellerkm(v,VUTNODER(v,a,k),k):=
-z(VUTNODER(v,a,k),k)+
M2(v,VUTNODER(v,a,k)) * x(v, VUTNODER(v,a,k),k)<=
M2(v,VUTNODER(v,a,k)) - KMTT(v,VUTNODER(v,a,k))-
LENGDE(VUTNODER(v,a,k))

end-if
end-do

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

if(PTOTAL=TTOTAL)then
TellerkmVlh(VINNODER(j,a,k),j,k):=
z(VINNODER(j,a,k),k) +
M3(VINNODER(j,a,k),j)*x(VINNODER(j,a,k),j,k)<=
M3(VINNODER(j,a,k),j)-KMTT(VINNODER(j,a,k),j)+ KMGRENSE(k)
end-if

if(PTOTAL>TTOTAL) then
    u:=j+ANTALLNODER
    TellerkmVlh(VINNODER(j,a,k),j,k):=
    z(VINNODER(j,a,k),k) +
M3(VINNODER(j,a,k),j)*x(VINNODER(j,a,k),j,k)<=
M3(VINNODER(j,a,k),j)-KMTT(VINNODER(j,a,k),j)+ KMGRENSE(k)

TellerkmVlh(VINNODER(u,a,k),u,k):=
z(VINNODER(u,a,k),k) +
M3(VINNODER(u,a,k),u)* x(VINNODER(u,a,k),u,k)<=
```

```
M3(VINNODER(u,a,k),u) - KMTT(VINNODER(u,a,k),u)+ KMGRENSE(k)
end-if
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..ADUTNODER(j,k)|
DAGSLOPSDAG(j)=1 and DAGSLOPSDAG(VUTNODER(j,a,k))=2)
x(j,DUTNODER(j,a,k),k)+

sum(j in NODER|j>ANTALLDAGSLOP and j<=ANTALLNODER,
a in 1..AVUTNODER(j,k)|DAGSLOPSDAG(j)=1 and
DAGSLOPSDAG(VUTNODER(j,a,k))=2) x(j, VUTNODER(j,a,k),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 and PTOTAL>TTOTAL) then

forall(j in NODER|j<=ANTALLDAGSLOP,k in TYPE,a in 1..ADINNODER(j,k)|
   DINNODER(j,a,k)<=ANTALLDAGSLOP or (DINNODER(j,a,k)>ANTALLNODER and
   DINNODER(j,a,k)<=ANTALLNODER+ANTALLDAGSLOP)and
   (DAGSLOPSDAG(j)<>1 and DAGSLOPSDAG(DINNODER(j,a,k))<>TTOTAL)) do

   NYj := j + ANTALLNODER
   VLHTALLNEDRE := ANTALLDAGSLOP+ANTALLNODER !konstant
   VLHTALLOVRE := ANTALLNODER+ANTALLNODER !konstant

   (x(DINNODER(j,a,k),j,k) +

   0.5* sum(b in 1..ADINNODER(j,k)|((DINNODER(j,b,k)>ANTALLDAGSLOP and
```

```
   DINNODER(j,b,k)<=ANTALLNODER) or (DINNODER(j,b,k)>VLHTALLNEDRE and
   DINNODER(j,b,k)<=VLHTALLOVRE)) and VUTNODER(DINNODER(j,a,k),b,k)=j)

   (x(DINNODER(j,a,k),DINNODER(j,b,k),k)+x(DINNODER(j,b,k),j,k)))=

   (x(DINNODER(NYj,a,k),NYj,k) +
   0.5*sum(b in 1..ADINNODER(NYj,k)|((DINNODER(NYj,b,k)>ANTALLDAGSLOP and
   DINNODER(NYj,b,k)<=ANTALLNODER) or (DINNODER(NYj,b,k)>VLHTALLNEDRE and
   DINNODER(NYj,b,k)<=VLHTALLOVRE)) and VUTNODER(DINNODER(NYj,a,k),b,k)=
     NYj)
   (x(DINNODER(NYj,a,k),DINNODER(NYj,b,k),k)+ x(DINNODER(NYj,b,k),NYj,k)))

end-do
end-if



!-----------------------------------------
!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)|PTOTAL>TTOTAL)
x(VINNODER(j+ANTALLNODER,a,k),j+ANTALLNODER,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)|
PTOTAL>TTOTAL)
KMTT(DINNODER(j+ANTALLNODER,a,k),j+ANTALLNODER)*
x(DINNODER(j+ANTALLNODER,a,k),j+ANTALLNODER,k)+

SUM(k in TYPE, j in NODER|j>ANTALLDAGSLOP and j<=ANTALLNODER,
a in 1..AVINNODER(j,k)|PTOTAL>TTOTAL)
KMTT(VINNODER(j+ANTALLNODER,a,k),j+ANTALLNODER)*
```

```
x(VINNODER(j+ANTALLNODER,a,k),j+ANTALLNODER,k)


!Vedlikehold er viktigst å minimere
Totalt := Vedlikehold + 1/100000*Tomtog

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

Totalt >= sum(i in NODER|i <= ANTALLDAGSLOP, k in TYPE)
(NODETYPE(i,k)* LENGDE(i) /KMGRENSE(k))*PTOTAL/TTOTAL


minimize(Totalt)


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

writeln('Antall vedlikehold er: ', getsol(Vedlikehold))
writeln(' ')
writeln('Antall tomtogskilometer er: ', getsol(Tomtog))
writeln(' ')
writeln('Totalt antall kilometer kjørt: ',
SUM(j in DAGSLOPSNODER)LENGDE(j)+getsol(Tomtog))
writeln(' ')


!skriver ut alle nodene som kobles sammen
forall (j in NODER|j <=ANTALLNODER, k in TYPE) do

if(j <=ANTALLDAGSLOP) then

forall (a in 1..ADINNODER(j,k)) do

if(getsol(x(DINNODER(j,a,k),j,k)) = 1)then
writeln ('Node ',DINNODER(j,a,k), ' og node ',j,' kobles sammen')
end-if

if(PTOTAL>TTOTAL) then
NYj := j + ANTALLNODER
if(getsol(x(DINNODER(NYj,a,k),NYj,k)) = 1)then
writeln ('Node ',DINNODER(NYj,a,k), ' og node ',NYj,
```

```
' kobles sammen')
end-if
end-if
end-do

elif(j>ANTALLDAGSLOP and j<=ANTALLNODER)then

forall(a in 1..AVINNODER(j,k)) do
if(getsol(x(VINNODER(j,a,k),j,k)) = 1)then
writeln ('Node ',VINNODER(j,a,k), ' og node ',j,' kobles sammen')
end-if
if(PTOTAL>TTOTAL) then
NYj := j + ANTALLNODER
if(getsol(x(VINNODER(NYj,a,k),NYj,k)) = 1)then
writeln ('Node ',VINNODER(NYj,a,k), ' og node ',NYj,
' kobles sammen')
end-if
end-if

end-do
end-if
end-do

writeln(' ')


if(PTOTAL=TTOTAL) then
ANTALLNODER2 := ANTALLNODER
 ANTALLDAGER := getsize(DAG)

elif(PTOTAL>TTOTAL) then
ANTALLDAGER := 2*getsize(DAG)
ANTALLNODER2 := getsize(NODER)
end-if


forall(s in STASJONER) do

writeln('Hensettingskapasiteten på stasjon ',s,' er: ',
 HKAP(s) )
forall(t in 1..ANTALLDAGER) do

forall(j in NODER|j<=ANTALLNODER2 and j>0 and (DAGSLOPSDAG(j)=t or
DAGSLOPSDAG(j)= t-TTOTAL)) do
```

```
if(j <= ANTALLDAGSLOP and j>0)then !j er dagslopsnode

forall(k in TYPE, a in 1..ADINNODER(j,k)|
(DAGSLOPSDAG(j) <> DAGSLOPSDAG(DINNODER(j,a,k)))) do

if (s = SLUTTSTED(DINNODER(j,a,k)) and
   getsol(x(DINNODER(j,a,k),j,k))=1) then

ANTALLTOG(s,t) +=1

end-if
end-do



elif(j>ANTALLDAGSLOP and j <=ANTALLNODER and
(DAGSLOPSDAG(j)=t or DAGSLOPSDAG(j)=t-TTOTAL))then

forall (k in TYPE, a in 1..AVINNODER(j,k)|
(DAGSLOPSDAG(j) <> DAGSLOPSDAG(VINNODER(j,a,k)))) do

if (s=SLUTTSTED(VINNODER(j,a,k))and
   getsol(x(VINNODER(j,a,k),j,k))=1) then

ANTALLTOG(s,t) +=1

end-if
end-do
end-if

end-do
writeln('Antall tog på stasjon: ',s,' på dag :',t,' er: ',
 ANTALLTOG(s,t))
end-do
end-do


end-model
```

Vedlegg C: Inndata til test

I dette vedlegget presenteres nodene i et av inndatasettene til modellen. På vedlagt cd ligger alle filene med inndatasettene, både som excelfiler og som datafiler.

På cd ligger også den implementerte modellen i en moselfil, i tillegg til løsninger etter å ha kjørt de ulike inndatasettene. Hvilket datasett som skal kjøres avhenger av hvilke parametre som er satt i koden som tidligere nevnt.

Tabell C.1: Inndata typer 1

Node	**NSBnr**	**Dag**	**Startsted**	**Sluttsted**	**Starttid**	**Sluttid**	**Lengde**
1	720002	1	KBG	KBG	05:55	23:44	927,48
2	720016	1	KBG	GUL	07:51	00:02	795,2
3	720004	2	KBG	EVL	04:55	01:12	1082,22
4	720006	2	KBG	KBG	06:53	01:51	711,35
5	720007	3	KBG	EVL	05:55	00:12	994,02
6	720003	3	EVL	DRM	05:24	21:36	819,25
7	720005	4	EVL	DRM	06:24	21:36	823,25
8	720016	4	KBG	GUL	07:51	00:02	795,2
9	720001	5	KBG	KBG	06:53	17:40	618,35
10	720006	5	KBG	KBG	06:53	01:51	711,35
11	720602	6	DRM	DRM	-	-	0
12	720601	6	DRM	DRM	-	-	0
13	720701	7	GUL	KBG	20:33	01:51	269,89
14	720702	7	DRM	DRM	-	-	0
15	-	1	SUND	SUND	03:00	07:00	-
16	-	1	SUND	SUND	23:00	03:00	-
17	-	2	SUND	SUND	03:00	07:00	-
18	-	2	SUND	SUND	23:00	03:00	-
19	-	3	SUND	SUND	03:00	07:00	-
20	-	3	SUND	SUND	23:00	03:00	-
21	-	4	SUND	SUND	03:00	07:00	-
22	-	4	SUND	SUND	23:00	03:00	-
23	-	5	SUND	SUND	03:00	07:00	-
24	-	5	SUND	SUND	23:00	03:00	-
25	-	6	SUND	SUND	11:00	15:00	-
26	-	6	SUND	SUND	15:00	19:00	-
27	-	7	SUND	SUND	11:00	15:00	-
28	-	7	SUND	SUND	15:00	19:00	-

(Tomtog på begynnelsen og slutten av dagsløpene er fjernet)