Vagtplanlægning med constraint programming

Christine Bliddal Ole Tranberg

oktober 2002

Forord

Denne rapport er resultatet af vores eksamensprojekt udarbejdet på Institut for Informatik og Matematisk Modellering (IMM) på Danmarks Tekniske Universitet. Projektet blev gennemført i perioden februar 2002 til oktober 2002 og repræsenterer 35 ECTS points.

Vores interesse for operationsanalyse samt ønsket om at få en praktisk betonet vinkel på vores arbejde udmundede i emnet vagtplanlægning på sygehuse ved brug af constraint programming.

Det forudsættes, at læseren har en generel viden indenfor operationsanalyse samt har kendskab til programmering.

Vi har i løbet af projektet fået mange gode råd og har haft mange interessante samtaler. I den forbindelse vil vi gerne rette en stor tak til vores vejleder Professor Jens Clausen (IMM) og vores kontakter på Carl Bro, Morten G. Madsen og Jakob Vestergaard Olsen. Endvidere vil vi gerne rette en varm tak til de personer i det danske sygehusvæsen, der tog sig tid til at snakke med os og besvare vores spørgsmål.

`Christine Bliddal`	`Ole Tranberg`

Oktober 2002.

Abstract

In this thesis we investigate the possiblities of using constraint programming in the area of staff planning by treating the nurse scheduling problem.

Based on interviews with several people in the hospital sector we create a mathematical model general enough to cover most of the common needs in this area.

The basics in constraint programming are introduced and we implement the previously developed model both in constraint programming and in linear programming.

For the implementation part we have used the constraint programming language ECL^{i}PS^e . The implementation is tested on problems inspired by the conditions met in the hospital wards and compared with an implementation in the more traditional programming environment GAMS.

We find that the constraint programming approach is a promising tool for staff scheduling problems since it allows for simpler implementation while offering potential for better run-time characteristics in terms of time and space used.

Keywords: staff planning, nurse scheduling, constraint programming, GAMS, ECL^{i}PS^e .

Resume

I dette eksamensprojekt undersøger vi mulighederne for at benytte constraint programming til løsning af vagtplanlægningsproblemer i praksis. Vi koncentrerer os om emnet omkring mandskabsplanlægning på sygehuse.

Med udgangspunkt i interviews med planlæggere fra forskellige hospitalsafdelinger udvikler vi en matematisk model, der er generel nok til at dække de fleste grundlæggende behov på dette område.

Vi giver en introduktion til det grundlæggende i constraint programming og implementerer den før definerede vagtplanlægningsmodel både ved hjælp af constraint programming og ved hjælp af lineær programmering.

Constraint programming modellen implementeres i ECL^{i}PS^e . Vi tester implementeringen på testeksempler inspireret af de forhold, vi stiftede bekendtskab med på hospitalsafdelingerne. Resultaterne fra constraint programming sammenlignes med resultaterne fra en lineær model implementeret i GAMS.

Vi konkluderer at constraint programming virker som et effektivt redskab til vagtplanlægningsproblemer. Det letter formuleringen både i overskuelighed og størrelse og giver bedre resultater i form af kortere køretider og bedre vagtplaner.

Nøgleord: vagtplanlægning, constraint programming, GAMS, ECL^{i}PS^e .

Indhold

Figurer

Kapitel 1

Indledning

Denne rapport handler om vagtplanlægning med constraint programming. Med udgangspunkt i et problem fra den virkelige verden undersøger vi mulighederne for at benytte constraint programming til at løse vagtplanlægningsproblemer.

Vagtplanlægning er et praktisk problem, der forekommer mange steder i erhvervslivet. At finde en løsning kan være yderst tidskrævende især på områder med behov for døgnbemanding. En løsning til et vagtplanlægningsproblem er en tildeling af personaleressourcer til ledige vagter, således at vagtbehovet bliver opfyldt. Jo flere individuelle ønsker der opfyldes, jo bedre er løsningen.

Der findes mange steder i erhvervslivet, hvor der er behov for døgnbemanding. Vi vil i denne rapport koncentrere os om sygehusafdelinger med døgnbemanding.

Arbejdstidsbestemmelser og upopulære vagter er nogle af de faktorer, der komplicerer vagtplanlægningsproblemet. Der er mange mulige vagtskemaer til hver ansat, og der er mange ansatte. I sidste ende skal summen af de individuelle vagtskemaer passe til det samlede bemandingsbehov.
Når man søger efter en løsning til en vagtplan, skal der foretages mange valg undervejs. De valg, man tager i starten af processen, har stor indflydelse på de senere muligheder.

Jo bedre planen bliver, jo mere tilfreds bliver personalet, og jo bedre bliver forholdene for de indlagte. Der er altså meget at vinde ved at lægge en god vagtplan.

Det er svært at lægge vagtplaner. En af udfordringerne er, at de involverer mennesker med individuelle præferencer. Dette gør det svært at udvikle en generel metode, der kan løse alle vagtplanlægningsproblemer.

Det tager ofte lang tid at lægge en vagtplan. En til to medarbejdere kan bruge op til to dage på at lægge en vagtplan, der gælder for 10-15 medarbejdere og strækker sig over 4 uger.

Et yderligere problem ved manuel vagtplanlægning er, at der kan forekomme ubevidst favorisering. Ud over at spare tid kan en automatisering af vagtplanlægningen derfor også udmunde i en mere retfærdig arbejdsfordeling.

Constraint programming

Constraint programming er en forholdsvis ny teknik indenfor operationsanalyse. Den udspringer fra kunstig intelligens og logikprogrammering og er især rettet mod kombinatoriske problemer, som vagtplanlægningsproblemet netop er.

Som for så mange andre programmeringsparadigmer er terminologien for constraint programming opstået på engelsk. Dette bevirker, at visse af termerne ikke findes på dansk; constraint programming er et eksempel herpå. Der findes ingen gode ord for constraint programming på dansk, hvorfor

vi i denne rapport ikke vil forsøge at oversætte begrebet men blot referere til det som constraint programming eller CP.

Flere forskellige sprog bygger ovenpå CP. Et af disse er ECL^{i}PS^e , som er et forholdsvis nyt sprog oprindelig udviklet af ECRC[1]. Videreudviklingen sker i dag på IC-parc[2].

1.1 Formål

Formålet med denne rapport er at undersøge egnetheden af CP i forbindelse med løsning af vagtplanlægningsproblemer samt at give en indsigt i vagtplanlægningsproblematikken for mandskab på sygehuse. Endvidere vil vi se på mulighederne inden for CP i forhold til almindelig lineær heltalsprogrammering.
Som holdepunkt i undersøgelsen vil vi udvikle et program, der er i stand til at løse vagtplanlægningsproblemer afgrænset af forholdene på sygehusafdelinger med døgnbemanding.

Til at udvikle programmet og udforske CP-mulighederne har vi benyttet programmeringssproget ECL^{i}PS^e .

[1] The European Computer- Industry Research Centre
[2] Centre for Planning and Resource Control at Imperial College in London

Kapitel 2

Samtaler

At bygge en god vagtplanlægningsmodel kræver viden om forholdene for det aktuelle arbejdsområde. For at opnå denne viden inden for sygehusvæsnet har vi studeret overenskomstreglerne og foretaget interviews med personale på flere sygehuse.

I forbindelse med disse interviews fik vi belyst hvilke forhold, der skal tages hensyn til og hvilke metoder, der benyttes i vagtplanlægningsprocessen.

Spørgsmålene til vores interviews og referater fra disse kan ses i bilag A.

Det skal bemærkes, at vi kun har snakket med dem, der lægger vagtplanerne. Vi har ikke interviewet nogen af dem, der "blot" følger planerne. Vi kender altså ikke medarbejdernes tilfredshed med de lagte planer. Dette kunne man følge op på med en undersøgelse af tilfredsheden med vagtplanerne blandt de ansatte. Dette ligger dog udenfor denne rapports område.

2.1 Sammendrag

Dette afsnit indeholder en sammenfatning af udbyttet fra vores interviews. Sammenfatningen kan ses som en opsummering af kravene fra den virkelige verden.

Vi har brugt det samme spørgeskema som udgangspunkt for alle samtalerne, dog kom vi ikke ind på alle punkter ved hver samtale. Alligevel har vi fået et ganske godt indblik i problematikken samt et overblik over hvilke ting, der skal tages hensyn til, når man skal lægge en vagtplan.

Ved hver samtale spurgte vi til slut, om der var emner, vi manglede at berøre. Ingen af de interviewede havde noget at tilføje her.
Dette skyldes formentlig, at vores samtaler hurtigt bredte sig ud over hele problematikken, hvorved vi også fik oplysninger om eventuelle problemstillinger, der ikke direkte var repræsenteret i vores spørgeskema.

Alt i alt mener vi, at vi har fået tilpas information om, hvordan vagtplanlægningen foregår i praksis. På dette grundlag vil vi bygge en model, der er i stand til at tage højde for mange relevante forhold i vagtplanlægningen.

Vi har haft samtaler med følgende:

Edel Sonne
Afdelingssygeplejerske på Gentofte Sygehus, Afdeling 437

Pia Varkill

Lægesekretær på Holbæk Sygehus, Skadestuen

Lisbet Pedersen
Afdelingssygeplejerske på Holbæk Sygehus, Afdeling A2

Cecilie Kabell
Lægesekretær på Hillerød Sygehus, Ortopædkirurgisk afdeling

Ole Rudkjøbing
Overlæge på Holbæk Sygehus

Nedenfor følger en uddybning af de væsentligste punkter fra vores samtaler. Til slut er en oversigt i tabel 2.1.

2.1.1 Overenskomster

Overenskomstreglerne skal i teorien altid overholdes. I praksis er der mange tilfælde, hvor reglerne omgås. Det er muligt for de enkelte faggrupper at lave lokalaftaler, der under visse forudsætninger giver lov til at bryde overenskomstreglerne. Reglerne er lavet for at beskytte personalet, og mange steder tager man mere hensyn til personalets ønsker end til overenskomsten.

2.1.2 Periodelængde

Der er ingen regel for, hvor lang tid en vagtplan skal gælde. Længden på vagtplanerne varierer meget fra sted til sted. Det mest almindelige er fire uger, men i perioder med meget ferie gælder planerne som regel for længere tid. På en enkelt af de afdelinger vi havde kontakt med, lægges planen kun for to uger ad gangen.

2.1.3 Rulleplaner

Problemet med at lægge en vagtplan gribes forskelligt an. Nogle steder starter man forfra ved hver vagtperiode og lægger en ny plan fra bunden af for den kommende periode. Andre steder arbejder man med rulleplaner, der kører i rul over perioderne, så man efter en vagtplansperiode starter forfra på planen.

I de tilfælde hvor der benyttes rulleplaner, fungerer denne primært som en skabelon for den nye vagtplan. Den kommende vagtplan tilpasses de ansattes fridagsønsker, ferie, kurser og lignende, og eventuelle ubesatte vagter besættes.

2.1.4 Fast behov

Almindeligvis har afdelingerne et uændret behov fra uge til uge. Behovet kan altså godt svinge fra dag til dag, men er det samme på for eksempel alle mandage. En afdeling har således et antal faste vagter, der skal dækkes. Det kan for eksempel være tre dagvagter fra 7 til 15 og en aftenvagt fra 15 til 23. Afdeling A2 på Holbæk Sygehus skiller sig dog ud. På denne afdeling opretter man nye vagter afhængig af patientbelægningen og timetallene for de enkelte medarbejdere. På denne måde får man en lang række af forskellige vagter med forskellige mødetider og længder. Dette gør, at det er meget svært at gennemskue, hvad minimumskravet til personaledækningen er.

De fleste steder er behovet mindre i weekenden. Der er altså færre på arbejde lørdage og søndage end på hverdage.

2.1.5 Fuld belægning

Det er interessant at vide, om alle vagter altid er besat i den færdige vagtplan, eller om der kan forekomme ubesatte vagter. Her er der ikke noget fællestræk for afdelingerne. På Hillerød Sygehus er der ikke personale nok, hvorfor der automatisk vil være ubesatte vagter. Disse vagter forsøger man at få besat ved at overtale folk til at tage overarbejde. Dette kan godt give en skæv arbejdsfordeling. På afdeling A2 på Holbæk Sygehus defineres vagterne i forhold til hvor meget personale, der er til rådighed. På denne måde forekommer der selvfølgelig ingen ubesatte vagter i vagtplanen. De andre steder passer personalemængden lige præcis til behovet. En færdig fuldt udfyldt vagtplan kan senere komme til at indeholde ubesatte vagter, eksempelvis ved sygdom. Selvom det hverken er ønskeligt eller optimalt, kan det til tider ske, at man er for få på arbejde i et tidsrum.

2.1.6 Hviletid mellem vagter

Arbejdsmiljølovens hviletidsregel [1] påbyder, at der skal gå mindst 11 timer, fra man får fri fra en vagt, til man skal møde på den næste. Flere steder er vi stødt på lokalaftaler, som giver lov til at bryde denne regel med op til tre timer dog kun et vist antal gange på fire uger. Et sted er det for eksempel muligt to gange på fire uger kun at have otte timer mellem to på hinanden følgende vagter.

2.1.7 Timetal

Det er ofte tilfældet, at medarbejderne er ansat på forskellige timetal. Det eneste sted af dem vi talte med, hvor alle de ansatte havde samme timetal, var hos lægerne på Holbæk Sygehus. Timetallene forsøges overholdt for hver uge, men dette er dog sjældent muligt. I stedet overholdes timetallene i gennemsnit over en periode på to til otte uger afhængig af vagtplanens længde.

2.1.8 Ens kvalifikationer

Personalet på en afdeling har ikke nødvendigvis de samme kvalifikationer. Der er forskel på at være social- og sundhedsassistent og på at være sygeplejerske, og derfor er der også forskel på, hvilke af medarbejderne man har på arbejde. De steder, hvor man både har social- og sundhedsassistenter (også kaldet sosu-assistenter) og sygeplejersker ansat, skal der altid være sygeplejersker på arbejde. For lægernes vedkommende er nogle uddannet til at kunne køre med læge-ambulance og andre ikke. Dette er en relevant faktor i vagtplanlægningen.

2.1.9 Nattevagter

Nattevagter er ofte upopulære blandt personalet. Nogle steder er der ansat fast personale til at tage nattevagter. Andre steder skal alle tage deres del af nattevagterne.
På Hillerød Sygehus undgår de så vidt muligt at give nattevagter til personer over 55 år, da det med alderen bliver sværere og sværere at vende døgnrytmen.
Nattevagterne bliver fordelt enten, så alle har lige mange nattevagter, eller så antallet af nattevagter er vægtet i forhold til den enkeltes timetal. Fælles for alle steder er, at hvis nogen specifikt ønsker nattevagter, får de det som regel uden problemer.

2.1.10 Fordeling af weekendvagter

Normalt arbejder afdelingerne ud fra, at alle medarbejderne tager weekendvagter. Weekendarbejde er et ansættelseskrav, og man ansættes til at arbejde hver anden eller hver tredje weekend. Formålet med kravet er at sørge for, at alle tager sin del af weekendvagterne. Når weekendarbejdet tildeles, gøres der meget ud af at sammenlægge weekendvagterne. På denne måde vil man ofte have samme type vagt lørdag og søndag, så medarbejderne enten arbejder både lørdag og søndag eller har fri hele weekenden. Formålet med dette er at kalde så få som muligt på arbejde i weekenderne.

2.1.11 Jul og nytår

Vi har set vagtfordelingen omkring jul og nytår løst på forskellige måder. Et sted holder de et fælles møde, hvor ingen går, før alle vagter er besatte. Et andet sted gives den enkelte medarbejder en valgprioritet samt et tilhørsforhold til enten jul eller nytår. Begge dele skifter fra år til år. Den med højeste prioritet vælger sin vagt først. Derefter vælger den med næsthøjeste prioritet og så fremdeles.

2.1.12 Sociale hensyn

På nogle afdelinger tages der hensyn til, om folk arbejder godt eller dårligt sammen. Her forsøger man at undgå at ansatte, der arbejder dårligt sammen, er på arbejde samtidig. På andre afdelinger forventes alle at kunne samarbejde.

2.1.13 Andet

Alle de steder, vi har haft kontakt med, foregår selve planlægningen manuelt. Planlæggerne begynder med at tildele weekend- og nattevagter til de ansatte. Derefter tildeles resten af vagterne.

På nogle afdelinger tages der hensyn til, om nogle vagttyper kan være uhensigtsmæssige at have efter hinanden. Dette kan både være et udtryk for en personalepolitik og for individuelle præferencer.

De fleste afdelinger vægter medarbejdernes ønsker om fridage højt. Under planlægningen efterkommes fridagsønskerne, hvis det på nogen måde er muligt.

2.2 Oversigt over resultater

I nedenstående tabel har vi sammenfattet resultaterne nævnt i forrige afsnit.

	E.S.	P.V.	L.P.	C.K.	O.R.
Periodelængde	4-8 uger	4 uger	2 uger	4 uger	4 uger
Rulleplan	nej	ja	ja	nej	nej
Fast ugentligt behov	ja	ja	nej	ja	ja
Fuld belægning	ja	nej	nej	ja	ja
Fast dagligt behov	ja	ja	nej	ja	ja
-mindre i weekenden	ja	nej	nej	ja	ja
Hviletid	8 (2 pr. md.)	11	11	8 (2 pr. md.)	11
Ansattes timetal	forskelligt	forskelligt	forskelligt	forskelligt	ens
-overholdes over	4-8 uger	4 uger	12 uger	2 uger	-
Ens kvalifikationer	nej	ja	nej	ja	nej
Fordeling af nattevagter	forskelligt	1 fast,ligeligt	alle	alle < 55 år	alle
Ford. af weekendvagter	alle	alle	alle	alle	alle
-sammenlægges	ja	ja	ja	ja	ja
Weekendarbejde	hver 3.	hver 2.	hver 3.	hver 2.	forskelligt
Fordeling jul/nytår	møde	efter tur	efter tur	møde	møde
Fridage pr. uge	-	1	1	2	-
Fridage pr. 14 dage	-	-	-	2 samlede	-
Fridagsønsker	ja	ja	ja	ja	nej
Sociale hensyn	nej	ja	nej	nej	nej

E.S.: Edel Sonne P.V.: Pia Varkill L.P.: Lisbet Pedersen
C.K.: Cecilie Kabel O.R.: Ole Rudkjøbing

Tabel 2.1: Sammenfatning af interviews

Kapitel 3

Afgrænsning

Det er altid svært at modellere virkeligheden. Matematiske modeller er meget firkantede i forhold til den virkelige verden. Derfor er man nødt til at lave en afgrænsning af sit problem og definere, hvilke forhold man vil medtage.

Vi tager udgangspunkt i forrige kapitels interviews og beslutter hvilke muligheder og hvilke forhold, der skal medtages i vagtplanlægningsmodellen.

Målet er ikke at udvikle en model, der inddrager alle forhold, men at lave en prototype der alligevel dækker de væsentligste aspekter i vagtplanlægningsproblemet. Først diskuterer vi relevansen af og muligheden for at medtage diverse emner, og til slut samler vi vores konklusioner i en oversigt over de endelige krav til vores model.

3.1 Diskussion

Som det fremgår af forrige kapitel, er der stor forskel på, hvordan planlægningen for personale på sygehuse foregår både med hensyn til regler og fremgangsmåde. Under beslutningsprocessen har vi forsøgt at gøre modellen så generel som muligt, så den kan tilpasses til flest mulige afdelinger. Samtidig har vi også haft for øje, at modellen gerne skulle være praktisk anvendelig.

3.1.1 Vagter

Behov og belægning

Vi vælger at lade behovet være fastlagt på forhånd, da det netop er tilfældet på de fleste sygehuse. Metoden med at definere behovet ud fra antallet af ansatte og antallet af patienter virker på os meget uhensigtsmæssig. Vi stiller det som en forudsætning, at man har et foruddefineret behov for hver dag, vagplanen dækker. Skulle programmet fastlægge personalebehovet fra dag til dag, ville det kræve yderligere information end den, vi har indsamlet fra vores samtaler.

En del af formålet med at benytte en matematisk model til at lægge en vagtplan er, at det gerne skulle give bedre vagtplaner til alle. Derfor skal vi tage stilling til, hvad der skal ske, hvis behov og ressourcer ikke passer sammen.

Hvis der er et større behov end personalets samlede timetal kan dække, kan man enten lade en dummy-vagt tage de overskydende vagter, eller prøve at fordele vagterne mellem personalet.
Dummy-vagten ændrer ikke meget på forholdene, som de er i dag, da lederne stadig skal ud og overtale medarbejderne til at tage ekstravagter. Vores hensigt er at lave et program, der kan fordele

vagterne blandt personalet. Derfor vil vi i modellen ikke tillade, at man sætter for få på arbejde.

Hvis behovet er mindre end personalets samlede timetal, kan man enten lade folk arbejde mindre, end de er ansat til, eller sætte flere folk på arbejde, end behovet foreskriver. Vi har valgt at benytte den første metode.

Bemandingsbehovet skal altså opfyldes præcist, og alle vagter skal altid besættes ved hjælp af det givne antal ansatte. Dette medfører, at det kan være nødvendigt med overarbejde. Senere i dette kapitel diskuterer vi, hvorledes eventuelt overarbejde skal fordeles.

Vagttyper

I forbindelse med vores samtaler har vi set flere forskellige slags vagttyper. Fælles for dem alle er, at de er delt op i tre kategorier: dag-, aften- og nattevagter. En afdeling har som regel to typer af dagvagter, en fra 7 til 15 og en fra 8 til 16. Ligeledes kan der også være forskellige aften- og nattevagter. Vi har valgt at operere med tre slags vagter. En for hver af de tre kategorier, dag, aften og nat.

Det ville naturligvis være optimalt, hvis modellen tog hensyn til flere vagttyper end disse tre. Da det alligevel ikke er muligt at tage højde for alle typer vagter på alle afdelinger, og en udvidelse til flere vagttyper har ringe praktisk betydning i forbindelse med at formulere og løse problemet, mener vi, at det er forsvarligt at indskrænke os til de tre nævnte vagttyper.

Hvis man ville tilføje flere vagter til vores model, ville disse skulle kædes sammen med de eksisterende vagter. Sammenkædningen ville foregå ved at tilføje en ekstra række af begrænsninger, som alle ville være af samme type som dem, vi i forvejen benytter. Dette ville ikke belyse nye interessante vinkler hverken i forbindelse med den matematiske formulering eller programmeringen i CP.

Ud over de nævnte tre vagttyper har vi valgt at benytte to passive vagttyper: sovevagt og frivagt. En sovevagt kan kun forekomme lige efter en eller flere nattevagter. Frivagten indikerer, at man har fri, og altså hverken er på arbejde eller har sovevagt. I det følgende vil vi også omtale sove- og frivagter som henholdsvis sovedage og fridage.

Vagtrækkefølge

På alle afdelingerne afsluttes en nattevagtsperiode med en sovedag. Dette er en ufravigelig regel, og vi føler det derfor uhensigtsmæssigt ikke at tage den med. Det vil sige, at efter en nattevagt må man kun have enten endnu en nattevagt eller en sovedag.

Periodelængde

Det er forskelligt fra afdeling til afdeling, hvor lang en periode der lægges vagtplan for. Dog er det fælles for alle afdelinger, at en ny vagtperiode altid starter på en mandag. Vi vil i første omgang sætte os det mål at lade en vagtplan gælde for fire uger.

Perioden skal altid starte på en mandag. I modellen betyder dette, at man meget let kan finde for eksempel alle lørdagene, da disse altid vil ligge på $7 \cdot i + 6,\ i \in \mathbb{N}_0$. På denne måde kan man let skille weekender fra hverdage.

3.1.2 Personale

Fast antal ansatte

Vi vil ikke fokusere på de økonomiske aspekter af vagtplanlægningen og dermed heller ikke medtage muligheden for at lade modellen justere på antallet af ansatte. I praksis arbejder planlæggerne ud fra et fast defineret antal ansatte og tildeler vagter til disse. De ser ikke på, om de kan klare sig med færre ansatte. På samme måde skal vores model tage udgangspunkt i et fast antal medarbejdere.

Som vi har set, er det forskelligt fra sted til sted, hvor meget personale der er i forhold til vagtbehovet. Nogle steder er de lige præcis så mange, som der er behov for, andre steder er de for lidt og klarer sig ved, at folk melder sig frivilligt til de sidste vagter. Som forudsætning for at lægge en vagtplan kræver vi, at der skal skal være personale nok til at dække behovet eventuelt ved brug af overarbejde.
Vi tager altså ikke højde for situationer, hvor der ikke er personale nok til at dække alle vagter, samtidig med at de fastsatte begrænsninger overholdes.

Kvalifikationer

Er det relevant at kunne angive hvilke kvalifikationer, de ansatte har? Der kan argumenteres både for og imod. Hovedsageligt mener vi, at det kommer an på hvilke slags kvalifikationer, det drejer sig om. Taler vi for eksempel om sygeplejersker kontra social- og sundhedsassistenter, er muligheden for at differentiere mellem dem i et vagtplanlægningsprogram ikke strengt nødvendig. Man kan klare sig ved at køre programmet med to sæt data, ét for sygeplejerskerne og ét for sosu-assistenterne. Taler man derimod om kvalifikationer i forlængelse af ens embede, såsom lægeambulancekvalifikationer, er sagen en anden. Her er de to grupper ikke adskilt, de overlapper blot hinanden. Det er altså ikke muligt at skille behovene og de ansatte i to separate grupper. Derfor finder vi det relevant at kunne skelne mellem personaleressourcerne.

For at vise mulighederne i at kunne differentiere mellem personalets forskellige kvalifikationer har vi valgt, at det skal være muligt at fritage nogle medarbejdere for nattevagter.

Timetal

Det er forskelligt, hvor meget hver medarbejder arbejder om ugen. Medarbejdere på samme afdeling kan godt være ansat på forskellige timetal. Da der er stor forskel på, om man arbejder 10 eller 25 timer om ugen, er det nødvendigt at kende de ansattes timetal, når man lægger en vagtplan. Derfor skal dette også være muligt i en vagtplanlægningsmodel. Når man definerer sit personale, skal man samtidig angive timetal for hver enkelt medarbejder.

3.1.3 Overenskomst og arbejdsmiljølov

En overenskomst dækker mange emner, og ikke alle er relevante i forbindelse med denne rapport. Vi vil derfor kun komme ind på dele af overenskomsten. Ønskes nærmere oplysning omkring overenskomsten henvises til [8]. Det mest interessante punkt i denne sammenhæng er arbejdstiden. Arbejdstiden beskrives i arbejdstidsaftaler [7] for de enkelte grupper. Heri er beskrevet krav til fritid, helligdage, overarbejde, tillæg og meget andet. Nedenfor nævner vi nogle af de væsentligste punkter.

Fridøgnsperioder

Ifølge arbejdstidsaftalens kapitel 6 §20 stk. 1 skal en ansat have en “ugentlig lang fridøgnsperiode af 55 til 64 timers varighed“. En lang fridøgnsperiode kan deles op i to mindre perioder på mindst 35

timer. Denne regel har vi tænkt os at overholde. På grund af fridøgnsperiodernes længde giver kravet visse begrænsninger på hvilke vagter, der kan ligge op til en fridøgnsperiode.

"Ugentlig" forstår vi her som faste uger fra mandag til mandag. For at overholde kravet om de 55 timer skal modellen sikre, at en medarbejder enten har fri to dage i træk, eller har kombinationen [dag,fri,nat].

[fri,fri] Hvis man har fri to dage i træk, vil disse tilsammen kun give en periode på 48 timer, hvilket er 7 timer mindre end det krævede. Det tidligste tidspunkt man derefter vil kunne møde, er på den følgende dagvagt. Da denne først begynder klokken 7, vil man i alt opnå et fridøgn på mindst 55 timer.

[dag,fri,nat] Kombinationen [dag,fri,nat] giver i alt 56 timers fri, da en dagvagt slutter klokken 15, og dermed giver 9 timer fri resten af dette døgn. En efterfølgende fridag giver 24 timer og nattevagtern herefter starter først klokken 23, hvilket giver yderligere 23 timer, i alt $9 + 24 + 23 = 56$ timers fri.

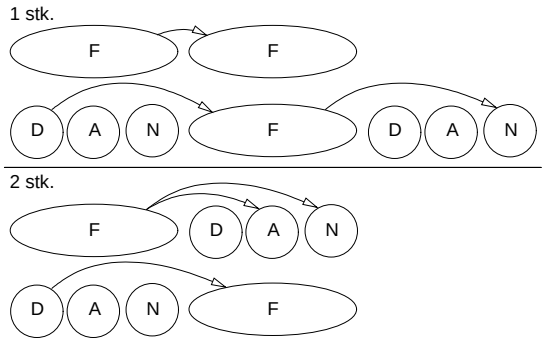
For at opnå et fridøgn på 35 timer kræves det, at medarbejderen har fri i et døgn, plus 11 timer. Dette er opfyldt, enten hvis en dagvagt efterfølges af en fridag, eller hvis en fridag efterfølges af en aftenvagt eller nattevagt.

[dag,fri] Dagvagten slutter klokken 15, hvilket giver i alt 9 timer til næste døgn, der er et fridøgn. Da man tidligst kan møde på dagvagt efter fridagen vil man få yderligere 7 timer fri. Dette giver i alt $9 + 24 + 7 = 40$ timer fri.

[fri,aften] Hvis man har aftenvagt efter en fridag, opnår man mindst $24 + 15 = 39$ timer fri.

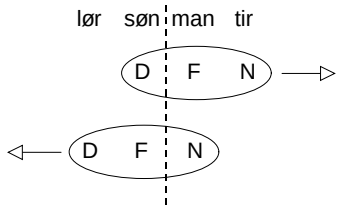
[fri,nat] En sidste mulighed for at have fri mere end 35 timer og mindre end 55 er, hvis man har nattevagt efter en fridag. Dette giver mindst $24 + 23 = 47$ timer fri.

Modellen skal altså sikre, at hver medarbejder i hver uge enten har fri to dage i træk, har kombinationen [dag,fri,nat] eller to gange har en af de tre vagtkombinationer: [dag,fri], [fri,aften], [fri,nat]. De forskellige muligheder er illustreret på figur 3.1, hvor pilene viser mulighederne for efterfølgende vagter i forbindelse med fridøgnsperioder.



Figur 3.1: Fridøgnsperioder

Op til et ugeskift, det vil sige ved overgangen fra søndag til mandag, definerer vi fridøgnsperioderne til at tælle med i den uge, frivagten er i. Kombinationen [fri,fri] tæller kun med, hvis den er helt indeholdt i den aktuelle uge. Fridøgnsperiodernes tilhørselsforhold er illustreret i figur 3.2.



Figur 3.2: Fridøgnsperioder ved ugeskift

Arbejde på særlige tidspunkter

Dette emne dækker mange aspekter. Både overtidsbetaling for vagter på hverdage udenfor almindelig arbejdstid og på søn- og helligdage. I arbejdstidsaftalens kapitel 3 står beskrevet de præcise tal. Det forholder sig sådan, at det ofte er hårdere at have aftenvagter og nattevagter frem for dagvagter. Derfor tæller aften- og nattevagter også som flere timer end dagvagter på trods af samme længde. I vores model tæller hver times arbejde mellem klokken 17 og 07, for 1,25 timer. Dette betyder, at en aftenvagt (15-23) kommer til at tælle for 9,5 timer, og at en nattevagt (23-07) tæller for 9,75 timer. Disse omregninger er baseret på arbejdstidsaftalens oplysninger.

Endvidere tæller hver vagt mere, når den ligger i weekenden. Dette fænomen vil vi ikke medtage, hvorfor lør-, søn- og helligdage ikke er længere end andre dage i vores model.

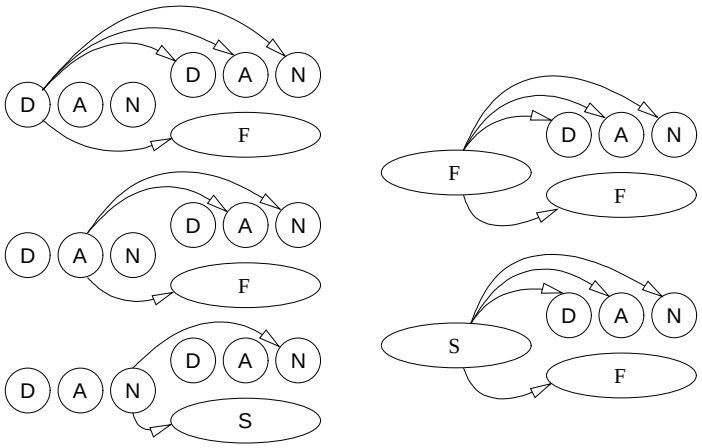
På mange afdelinger har de en fast politik om, at man skal have fri hver anden weekend. Denne restriktion har vi valgt at tage med i vores model

11-timersreglen

Arbejdsmiljølovens bestemmelse om 11 timers hviletid mellem to på hinanden følgende vagter vil vi gerne medtage i vores model. Denne lov har stor indflydelse på, hvordan ens vagtskema kan se ud. Derfor vil det ikke være hensigtsmæssigt at se bort fra denne.

På flere afdelinger har de lokalaftaler, der gør det muligt at nedsætte hviletiden til 8 timer for eksempel to gange på en måned. Ikke alle afdelinger har lokalaftaler, og lokalaftalerne behøver heller ikke at være ens fra sted til sted. Vi har valgt kun at lade 11-timers reglen gælde.

Kravene vedrørende nattevagter, sovedage og 11-timersreglen medfører, at nogle kombinationer af vagtrækkefølger ikke må være mulige i modellen. Hvis man har dagvagt efter en aftenvagt, er hviletidsreglen ikke overholdt. Dette skal altså ikke være muligt. I figur 3.3 har vi vist de mulige vagtkombinationer for to på hinanden følgende dage.



Figur 3.3: Efterfølgende vagter

3.1.4 Personlige hensyn

Fridagsønsker

Da personalets arbejdstider varierer meget, er det relevant at inddrage muligheden for ønsker om fridage.

Det er tænkeligt, at man nogle gange kun har brug for at få fri en aften eller en eftermiddag og på denne måde godt kan arbejde de to andre vagter den dag. Muligheden for at ønske fri en enkelt vagt ville give større fleksibilitet i vagtplanlægningen. Dog ville det være upraktisk, at skulle ønske fri tre gange for at få fri en hel dag, for at undgå dette kunne man lave flere typer af fridagsønsker.

For at vise mulighederne i forbindelse med fridagsønsker har vi valgt at inddrage muligheden for at ønske fri en hel dag ad gangen.

Det kan give problemer at ønske fri flere dage i træk, da alle ønsker ikke nødvendigvis kan opfyldes. Hvis man ikke kan få fri alle dage, er det ikke sikkert at man overhovedet er interesseret i at få fri de andre dage.

Ønsket om fridage vil vi modellere ved at angive et tal for hver medarbejder for hver dag. Er tallets værdi større end nul, ønsker medarbejderen fri, jo større tal, jo større ønske og jo større pris, hvis man ikke får fri.

Sygdom og ferie

Vi vil ikke medtage muligheden for at lade modellen lægge ferierne for de ansatte. For det første skal ferierne planlægges mere end én vagtplan forud, og for det andet er det ikke sikkert, at der er faste regler for, hvordan den fraværendes arbejde dækkes. Vi vil ligeledes heller ikke komme ind på problematikken omkring sygemeldinger i vores model. Enten må man tage den syge medarbejder ud af medarbejderstyrken og så lægge vagtplanen, eller også må man benytte en vikar.

Samarbejde

Af de fem afdelinger vi besøgte, er det kun på den ene, de tager specielle hensyn til, om de ansatte er i stand til at arbejde sammen eller ej. Den generelle holdning er, at alle skal kunne arbejde sammen.

Vi finder ikke, at hensynet til samarbejde vejer tungt i problematikken med at lægge en vagtplan. Derfor har vi ikke tænkt os at medtage dette i modellen.

3.1.5 Vagtfordeling

Et tilbagevendende spørgsmål er, hvordan vagterne fordeles retfærdigt. Hvordan bestemmes det, hvem, der skal have aftenvagt og hvem, der skal have dagvagt? Vi finder det mest retfærdigt at gå objektivt frem efter en slags matematisk retfærdighed. Alle kvalificerede skal tage deres del af nattevagterne, og derefter skal alle tage deres del at af aften- og dagvagter.

Vi har valgt at gå frem efter timetallene, så man får sin del af vagter svarende til ens timetal i forhold til det samlede timetal.

Overarbejde

Overarbejde kan fordeles på forskellige måder. Man kan for eksempel fordele overarbejdet ligeligt blandt de ansatte, så alle får lige mange overarbejdstimer over et givet stykke tid. En anden mulighed er at fordele overarbejdet relativt, set i forhold til hvor mange timer man er ansat til. I forbindelse med vores interviews så vi begge metoder benyttet.

Vi har valgt at gå frem efter den sidste metode med at fordele overarbejdet i forhold til de ansattes timetal. På denne måde får de, der er ansat til færrest timer også mindst overarbejde.
Dette har den konsekvens, at ved overarbejde får de med det største timetal flere overarbejdstimer end de med lille timetal. Omvendt kan det være uhensigtsmæssigt, hvis de med lave timetal risikerer at få dobbelt så meget arbejde, som de er ansat til.

Hvis det er tilfældet, at det samlede behov er mindre end personalets samlede timetal, behandles dette som "negativt" overarbejde. Her vil der altså også ske en relativ fordeling af "underarbejdet".

På denne måde er det ikke sikkert, at de ansatte får tildelt lige så mange timer, som de er ansat til. Dette ser vi ikke som det store problem. For det første er det jo altid muligt at tildele ekstravagter manuelt, hvis underarbejdet er et problem. For det andet har vi endnu ikke stødt på noget sted, hvor de har et behov for færre antal timer, end de har personale til.

Nattevagter

Da det ikke nødvendigvis er alle, der skal have nattevagter, ser vi først på fordelingen af disse. Vi lader $n_s = 1$ betegne, at sygeplejerske s kan have nattevagter og $n_s = 0$, at hun ikke kan. Hver sygeplejerske s har timetallet h_s. Andelen af nattevagter til sygeplejersken vil således være:

$$\frac{n_s \cdot h_s}{\sum_{i=1}^{S} n_i \cdot h_i} \tag{3.1}$$

Har man for eksempel 4 sygeplejersker, hvoraf 3 kan tage nattevagter ,og har de alle samme timetal, lad os sige 12 timer, vil andelene af nattevagter til de sygeplejersker, der kan have nattevagter være:

$$\frac{1 \cdot 12}{1 \cdot 12 + 1 \cdot 12 + 1 \cdot 12 + 0 \cdot 12} = \frac{1}{3} \tag{3.2}$$

Hvis det samlede behov for nattevagter er B_n, vil sygeplejserske s altså skulle have følgende antal nattevagter:

$$\frac{n_s \cdot h_s}{\sum_{i=1}^{S} n_i \cdot h_i} \cdot B_n = X_{s,nat} \tag{3.3}$$

Dette tal er ikke nødvendigvis et heltal. Da man ikke kan have andet end hele vagter, runder vi tallet af og finder derfor et interval for antallet af nattevagter for sygeplejerske s:

$$[\lceil X_{s,nat} \rceil - 1; \lceil X_{s,nat} \rceil] \tag{3.4}$$

Grænserne for antallet af nattevagter behandler vi som hårde grænser, sådan at forstå at det ikke er tilladt at overskride dem.

Dag- og aftenvagter

Dag- og aftenvagterne fordeles også i forhold til timetallene. Dog kan man ikke benytte de samme timetal som før, da nogle af timerne allerede er brugt til nattevagter. Vi justerer derfor timetallene, ved at trække de timer fra, der er brugt til nattevagter. Betegner vi det justerede timetal h'_s, skal sygeplejerske s altså have følgende antal af vagttypen a:

$$\frac{h'_s}{\sum_{i=1}^{S} n_i \cdot h'_i} \cdot B_a = X_{s,a} \ , \tag{3.5}$$

hvor a betegner hhv. dag- og aftenvagter.

Ligesom ved nattevagter er det ikke sikkert, at $X_{s,a}$ er et heltal, og vi runder igen tallet af, hvorved vi får et interval for antallet af vagter af typen a til sygeplejerske s:

$$[\lceil X_{s,a} \rceil - 1; \lceil X_{s,a} \rceil] \tag{3.6}$$

Disse grænser behandles som bløde, og en overtrædelse af intervallet vil tilføre en positiv værdi til objektfunktionen.

Weekendvagter

Folk vil som regel hellere have fri end være på arbejde i weekenden. Derfor sørger planlæggerne for, at så få som muligt bliver kaldt på arbejde i weekenden. I praksis betyder dette, at hvis en medarbejder skal på arbejde lørdag, skal han det også om søndagen, og som regel har han også samme type vagt de to dage. Da dette ikke altid kan lade sig gøre, har vi valgt at tage det med i vores model som en blød begrænsning. Modellen vil forsøge at samle weekenderne, men det vil være muligt at lave en vagtplan, hvor det ikke altid er tilfældet. Da man ikke kan have to sovedage i træk, vil vi også betragte kombinationen [sovedag,fri] gratis i forbindelse med sammenkædning af weekendvagter.

3.1.6 Generelt

En vagt pr. dag

På grund af 11-timersreglen er det ikke muligt for en medarbejder at have mere end en vagt pr. døgn. Da hver vagttype varer otte timer, vil der højst kunne være otte timer mellem to vagttyper, der hører inden for samme døgn, når man arbejder med de før definerede vagter.
Dette forhold betyder, at vi direkte kan modellere modellen således, at hver medarbejder skal have netop en af de 5 vagttyper for hver dag.

Vagtrækkefølge ([sove,nat])

Det er interessant at se på muligheden af at kunne definere gode og dårlige vagtrækkefølger. Som et eksempel vil vi medtage muligheden for at knytte en pris til vagtrækkefølgen sovedag efterfulgt

af nattevagt. Ved en sådan rækkefølge vil man have nattevagt, sovedag og derefter nattevagt igen. Denne rækkefølge er meget forstyrrende dels for medarbejdernes privatliv, dels for hensynstagen til deres søvnbehov. Derfor skal modellen sørge for at tilføje en pris til den samlede omkostning, hvis denne kombination er tilfældet.

Jul og nytår

Vores opfattelse er, at fordelingen af arbejdsvagter jul og nytår er et meget følsomt emne. Der er mange personlige forhold og interne retningslinjer, så det at lave et program der automatisk kan tildele jule- og nytårsvagter i alle tilfælde, fremstår som en umulig opgave. Vi vil derfor slet ikke inddrage denne problematik i vores model.

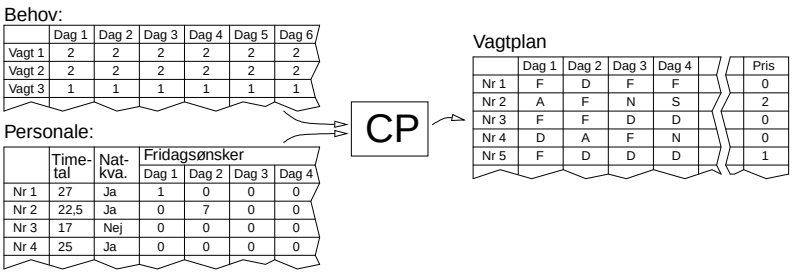
Overførsel fra periode til periode

For at opretholde en løbende sammenhæng mellem vagterne sørger planlæggerne for at sammenholde de sidste dage i den forrige vagtplan med starten af den nye vagtplan.

Dette har vi valgt ikke at tage med i vores model, da vi ikke finder det relevant for målet med dette projekt. I praksis vil det dog ikke være noget problem at udbygge modellen til også at tage højde for dette. I afsnit 6.5 fortæller vi om mulighederne for at indbygge dette.

3.2 Endelige krav

Den endelig model skal være i stand til at finde en vagtplan, givet en række behov og en personalemængde, og den færdige vagtplan skal overholde de fastsatte begrænsninger.



Figur 3.4: Endelig model

3.2.1 Inddata

Inddata til modellen består af:

Behov
: Bemandingsbehov for hver arbejdstype for hver dag.

Personale
: Information om hver medarbejders *timetal*, *natkvalifikation* og *fridagsønsker*.

3.2.2 Uddata

Modellen skal levere:

Vagtplan
Oversigt over hvem der skal arbejde hvornår.
Samlet omkostning
Pris for hvor god den fundne løsning er.

3.2.3 Hårde begrænsninger

De hårde begrænsninger er de begrænsninger, der altid skal være overholdt.

En vagttype pr. dag
Hver medarbejder skal have præcis en vagttype pr. dag.

Behovet opfyldes præcist
Der skal altid være det angivne antal medarbejdere på vagt.

Nattevagter
Det skal være muligt at fritage medarbejdere for nattevagter.

Sovedagsrestriktioner
En følge af en eller flere nattevagter skal altid afsluttes med en sovedag. Sovedage findes kun efter nattevagter.

11-timersreglen
Der skal mindst være 11 timer mellem to på hinanden følgende vagter.

Fridøgnsperioder
For hver uge skal hver medarbejder enten have en fridøgnsperiode på mindst 55 timer eller to fridøgnsperioder på mindst 35 timer.

En fridag pr. 7 dage
Hver medarbejder skal have mindst en fridag for hver 7 på hinanden følgende dage.

Fri hver anden weekend
Hver medarbejder skal have fri mindst hver anden weekend.

3.2.4 Bløde begrænsninger

De bløde begrænsninger er det tilladt at overtræde, men hvis dette sker tiløjer overtrædelsen en pris til objektfunktionen. Det skal være muligt at vægte de bløde begrænsninger i forhold til hinanden. Derfor skal der defineres en pris for hver begrænsning.

Vagter deles ligeligt
Antallet af vagter skal deles ligeligt imellem de ansatte, idet der tages højde for disses timetal.

Fridagsønsker
Det skal være muligt for medarbejderne at ønske fri på bestemte dage.

Weekendvagter
Hver medarbejder skal helst kun have en type vagt i hver weekend.

Vagtrækkefølge
Det er ikke ønskeligt at have nattevagt efter en sovedag.

Kapitel 4

Matematisk model

I dette kapitel opstiller vi en matematisk model, der følger retningslinjerne for vagtplanlægningsproblemet afgrænset i forrige kapitel. Modellen opbygges med henblik på at blive løst ved hjælp af en lineær solver i GAMS.

Først præsenterer vi den benyttede notation, og derefter beskriver vi, hvordan begrænsningerne modelleres. Til sidst samler vi det hele til en egentlig matematisk model.

Det er hensigten, at modellen skal dække de samme aspekter som den CP-model, vi senere vil konstruere.

4.1 Notation

Den notation, vi benytter i den matematiske model, er beskrevet herunder. Nogle af parametrene er af hensyn til modelleringen allerede fastlagt, mens andre blot forudsættes kendt som input.

Vi opererer med fem forskellige vagttyper ($t = 1 \ldots 5$). Det skal bemærkes, at kun tre af disse er egentlige arbejdsvagter. Disse tre har vi valgt at kalde for *arbejdstyper* ($a = 1 \ldots 3$). Når der defineres et behov til vagtplanen, er det altså kun i forbindelse med de første tre vagttyper.

De fem vagttyper er:

1 : Dagvagt	$(t = 1, a = 1)$	4 : Sovedag	$(t = 4)$
2 : Aftenvagt	$(t = 2, a = 2)$	5 : Fridag	$(t = 5)$
3 : Nattevagt	$(t = 3, a = 3)$		

Parametre

$S \in \mathbb{N}$	Antal sygeplejersker til rådighed.
$D \in \mathbb{N}$	Antal dage vagtplanen lægges for ($D = 28$).
$T \in \mathbb{N}$	Antal forskellige vagttyper ($T = 5$).
$A \in \mathbb{N}$	Antal forskellige arbejdstyper ($A = 3$).
$b_{d,a} \in \mathbb{N}_0$	Behov for arbejdstype a på dag d.
$o_{d,s} \in \mathbb{N}_0$	Fridagsønskeprioritet for sygeplejerske s, dag d.
$h_s \in \mathbb{N}$	Timetal for sygeplejerske s.
$n_s \in \{0, 1\}$	Natkvalifikation for sygeplejerske s. $n_s = 1$ angiver, at medarbejderen kan have nattevagt, og $n_s = 0$ angiver det modsatte.
$p_i \in \mathbb{N}_0$	Priser der fastsætter, hvor meget en overtrædelse af en blød begrænsning bidrager med i objektfunktionen ($i = 1, \ldots, 4$).
$NDel_s$	Andel af nattevagter sygeplejerske s skal have.
$NMax_s$	Det største antal nattevagter sygeplejerske s må have.
$NMin_s$	Det mindste antal nattevagter sygeplejerske s må have.
$DADel_s$	Andel af dag- og aftenvagter sygeplejerske s skal have.
$DMax_s$	Blød øvre grænse for antallet af dagvagter til sygeplejerske s.
$DMin_s$	Blød nedre grænse for antallet af dagvagter til sygeplejerske s.
$AMax_s$	Blød øvre grænse for antallet af aftenvagter til sygeplejerske s.
$AMin_s$	Blød nedre grænse for antallet af aftenvagter til sygeplejerske s.

For ovenstående gælder:

$$
\begin{aligned}
t \in \{1, \ldots, T\} &= \mathbf{T} \\
a \in \{1, \ldots, A\} &= \mathbf{A} \subseteq \mathbf{T} \\
d \in \{1, \ldots, D\} &= \mathbf{D} \\
s \in \{1, \ldots, S\} &= \mathbf{S}
\end{aligned}
$$

Beslutningsvariable

Modellen indeholder følgende beslutningsvariable, der alle beskrives i de næste afsnit:

$$
\begin{aligned}
x_{d,t,s} &\in \{0,1\} && s \in \mathbf{S},\ d \in \mathbf{D}, t \in \mathbf{T} \\
ff_{d,s} &\in \{0,1\} && s \in \mathbf{S},\ d \in \mathbf{D} \setminus \{D\} \\
dfn_{d,s} &\in \{0,1\} && s \in \mathbf{S},\ d \in \mathbf{D} \setminus \{D-1, D\} \\
dfd_{d,s} &\in \{0,1\} && s \in \mathbf{S},\ d \in \mathbf{D} \setminus \{D\} \\
dfa_{d,s} &\in \{0,1\} && s \in \mathbf{S},\ d \in \mathbf{D} \setminus \{D\} \\
afa_{d,s} &\in \{0,1\} && s \in \mathbf{S},\ d \in \mathbf{D} \setminus \{D\} \\
afn_{d,s} &\in \{0,1\} && s \in \mathbf{S},\ d \in \mathbf{D} \setminus \{D\} \\
sfa_{d,s} &\in \{0,1\} && s \in \mathbf{S},\ d \in \mathbf{D} \setminus \{D\} \\
sfn_{d,s} &\in \{0,1\} && s \in \mathbf{S},\ d \in \mathbf{D} \setminus \{D\} \\
u2ff_s &\in \mathbb{N}_0 && s \in \mathbf{S} \\
u2dfn_s &\in \mathbb{N}_0 && s \in \mathbf{S} \\
u2dfd_s &\in \mathbb{N}_0 && s \in \mathbf{S} \\
u2dfa_s &\in \mathbb{N}_0 && s \in \mathbf{S} \\
u2afa_s &\in \mathbb{N}_0 && s \in \mathbf{S} \\
u2afn_s &\in \mathbb{N}_0 && s \in \mathbf{S} \\
u2sfa_s &\in \mathbb{N}_0 && s \in \mathbf{S} \\
u2sfn_s &\in \mathbb{N}_0 && s \in \mathbf{S} \\
DMaxVar_s &\in \mathbb{N}_0 && s \in \mathbf{S} \\
DMinVar_s &\in \mathbb{N}_0 && s \in \mathbf{S} \\
AMaxVar_s &\in \mathbb{N}_0 && s \in \mathbf{S} \\
AMinVar_s &\in \mathbb{N}_0 && s \in \mathbf{S} \\
bweek_{i,u,s} &\in \{0,1\} && s \in \mathbf{S},\ i \in \{1,2\},\ u \in \{1, \ldots, \tfrac{D}{7} - 1\} \\
bens_{t,u,s} &\in \{0,1\} && s \in \mathbf{S},\ t \in \{1, \ldots, T\},\ u \in \{1, \ldots, \tfrac{D}{7}\} \\
bv_{d,s} &\in \{0,1\} && s \in \mathbf{S},\ d \in \mathbf{D} \setminus \{D\}
\end{aligned}
$$

Her er $x_{d,t,s}$ tildeling af vagttype t til sygeplejerske s, dag d. $x_{d,t,s} = 1$ angiver at der arbejdes på vagten, og 0 at der ikke gør.

De binære variable $ff_{d,s}$, $dfn_{d,s}$, $dfd_{d,s}$, $dfa_{d,s}$, $afa_{d,s}$, $afn_{d,s}$, $sfa_{d,s}$ og $sfn_{d,s}$ repræsenterer hver deres vagtrækkefølge. Her står d for dag, a for aften og så fremdeles. En værdi på 1 angiver, at vagtrækkefølgen er opfyldt for dag d. For eksempel angiver $afa_{d,s} = 1$, at sygeplejerske s har aftenvagt dag d, fri dag $d+1$ og aftenvagt igen dag $d+2$.
$u2ff$, $u2dfn$, $u2dfd$, $u2dfa$, $u2afa$, $u2afn$, $u2sfa$ og $u2sfn$ er antallet af føromtalte vagtrækkefølger i uge to. Tilsvarende findes alle disse variable også for uge 1, uge 3 og uge 4. Disse er ikke medtaget her, men kan ses i GAMS-koden i bilag C.2.

$DMaxVar_s$, $DMinVar_s$, $AMaxVar_s$ og $DMinVar_s$ benyttes i forbindelse med fordelingen af antallet af dag- og aftenvagter og angiver antallet af vagter, man har for meget henholdsvis for lidt i forhold til de bløde grænser.

De binære variable $bweek_{i,u,s}$, $bens_{t,u,s}$ og $bv_{d,s}$ benytter vi til at sikre, at mindst et af flere krav er opfyldt. Den nærmere brug er beskrevet første gang, vi benytter $bweek$ under kravet om friweekender.

4.2 Hårde begrænsninger

I det følgende gennemgår vi de begrænsninger, der altid skal overholdes.

Behovet opfyldes præcist

Behovet skal altid være opfyldt. Der må ikke være for få og ikke for mange på arbejde men præcist det antal, som der er behov for.

$$\sum_{s=1}^{S} x_{d,a,s} = b_{d,a}, \;\; d \in \mathbf{D}, a \in \mathbf{A} \tag{4.1}$$

En vagttype pr. dag

På grund af hviletidsreglen og arbejdstiderne er det kun muligt at have en vagt pr. dag.

$$\sum_{t=1}^{T} x_{d,t,s} = 1, \;\; d \in \mathbf{D}, s \in \mathbf{S} \tag{4.2}$$

Nattevagter

Hvis en medarbejder er fritaget for nattevagter, skal modellen sørge for, at vedkommende ikke får tildelt nattevagter. Hertil benyttes parameteren $n_s \in \{0, 1\}$, der indikerer, om sygeplejerske s må have nattevagter eller ej. $n_s = 0 \Rightarrow$
$\sum_{d=1}^{D} x_{d,3,s} = 0, s \in \mathbf{S}$.

$$x_{d,3,s} \leq n_s, \;\; d \in \mathbf{D}, s \in \mathbf{S} \tag{4.3}$$

Fridøgnsperioder

Som beskrevet i kapitel 3 skal man hver uge enten have en lang sammenhængende fridøgnsperiode eller to mindre. Der findes to muligheder for lange fridøgnsperioder ([frivagt,frivagt] og [dagvagt,frivagt,nattevagt]). Vi knytter en binær variabel til hvert af tilfældene til sygeplejerske s for hver af dagene. Den binære variabel tvinger tilfældet til at være sandt, hvis den antager værdien 1.

$$x_{d,5,s} + x_{d+1,5,s} \geq 2 \cdot ff_{d,s}, \;\; d \in \mathbf{D} \setminus \{D\}, s \in \mathbf{S} \tag{4.4}$$

$$x_{d,1,s} + x_{d+1,5,s} + x_{d+2,3,s} \geq 3 \cdot dfn_{d,s}, d \in \mathbf{D} \setminus \{D\}, s \in \mathbf{S} \tag{4.5}$$

Der findes 6 muligheder for korte fridøgnsperioder ([dagvagt,frivagt,dagvagt], [dagvagt,frivagt,aftenvagt], [aftenvagt,frivagt,aftenvagt], [aftenvagt,frivagt,nattevagt], [sovevagt,frivagt,aftenvagt] og [sovevagt,frivagt,nattevagt]). Først opretter vi en binær variabel for hvert af tilfældene for hver dag, fridøgnsperioden kan begynde på. Hvis variablen er lig 1, tvinges den tilhørende fridøgnsperiode til at være opfyldt. Dette er ikke tilfældet hvis den binære variabel er lig 0.

$$x_{d,1,s} + x_{d+1,5,s} + x_{d+2,1,s} \geq 3 \cdot dfd_{d,s} \tag{4.6}$$

$$x_{d,1,s} + x_{d+1,5,s} + x_{d+2,2,s} \geq 3 \cdot dfa_{d,s} \tag{4.7}$$

$$x_{d,2,s} + x_{d+1,5,s} + x_{d+2,2,s} \geq 3 \cdot afa_{d,s} \tag{4.8}$$

$$x_{d,2,s} + x_{d+1,5,s} + x_{d+2,3,s} \geq 3 \cdot afn_{d,s} \tag{4.9}$$

$$x_{d,4,s} + x_{d+1,5,s} + x_{d+2,2,s} \geq 3 \cdot sfa_{d,s} \tag{4.10}$$

$$x_{d,4,s} + x_{d+1,5,s} + x_{d+2,3,s} \geq 3 \cdot sfn_{d,s} \tag{4.11}$$

For (4.6)-(4.11) gælder $d \in \mathbf{D} \setminus \{D\}, s \in \mathbf{S}$

Fridøgnsperioderne skal knyttes til hver uge. Dette gør vi ved at summere ovenstående variable for dagene i ugen. Tilhørsforholdene ved ugeskift er som beskrevet i kapitel 3. På denne måde får vi en

binær variabel for hver fridøgnsperiode i hver uge. Dette er ret omfattende og giver mange variable, og vi viser det kun for uge nummer 2 i ligning (4.12) - (4.19). Resten af ugerne modelleres tilsvarende, blot skal man huske at ændre summationsparametrene. I første og sidste uge af vagtplanens periode skal henholdsvis starten af ugen og slutningen af ugen behandles som specialtilfælde, idet de ikke hænger sammen med henholdsvis foregående og efterfølgende uge. Udtrykkene for alle ugerne er beskrevet i GAMS-koden i bilag C.2.

Uge nummer 2 starter mandag $d = 8$ og slutter søndag $d = 14$.

$$u2ff_s = \sum_{d=8}^{13} ff_{d,s}, \quad s \in \mathbf{S} \tag{4.12}$$

$$u2dfn_s = \sum_{d=7}^{13} dfn_{d,s}, \quad s \in \mathbf{S} \tag{4.13}$$

$$u2dfd_s = \sum_{d=7}^{13} dfd_{d,s}, \quad s \in \mathbf{S} \tag{4.14}$$

$$u2dfa_s = \sum_{d=7}^{13} dfa_{d,s}, \quad s \in \mathbf{S} \tag{4.15}$$

$$u2afa_s = \sum_{d=7}^{13} afa_{d,s}, \quad s \in \mathbf{S} \tag{4.16}$$

$$u2afn_s = \sum_{d=7}^{13} afn_{d,s}, \quad s \in \mathbf{S} \tag{4.17}$$

$$u2sfa_s = \sum_{d=7}^{13} sfa_{d,s}, \quad s \in \mathbf{S} \tag{4.18}$$

$$u2sfn_s = \sum_{d=7}^{13} sfn_{d,s}, \quad s \in \mathbf{S} \tag{4.19}$$

Nu kan vi udtrykke fridøgnskravene for hver uge for hver sygeplejerske. Fridøgnskravene for uge 2 ser ud som beskrevet i ligning (4.20). En lang fridøgnsperiode tæller som to korte.

$$2 \cdot u2ff_s + 2 \cdot u2dfn_s + u2dfd_s + u2dfa_s + u2afa_s + u2afn_s + u2sfa_s + u2sfn_s \geq 2, \; s \in \mathbf{S} \tag{4.20}$$

Fri hver anden weekend

Hver medarbejder skal have fri mindst hver anden weekend. Til hjælp i denne begrænsning benytter vi de binære variable $bweek_{1,u,s}$ og $bweek_{2,u,s}$. Variablene repræsenterer tilfældene at sygeplejerske s har fri weekend u henholdsvis weekend $u + 1$. Mindst en af variablene skal have værdien en, før kravet er opfyldt. For at have fri en weekend skal sygeplejerske s have frivagt både lørdag og søndag, det vil sige $x_{d,5,s} = x_{d+1,5,s} = 1$, hvor d svarer til en lørdag. Det at have fri en weekend svarer altså til at summen af de to variable er 2. For at opfylde et af tilfældene benytter vi følgende ligninger

$$x_{7 \cdot u-1,5,s} + x_{7 \cdot u,5,s} \geq 2 \cdot bweek_{1,u,s} \tag{4.21}$$

$$x_{7 \cdot u+6,5,s} + x_{7 \cdot u+7,5,s} \geq 2 \cdot bweek_{2,u,s} \tag{4.22}$$

$$bweek_{1,u,s} + bweek_{2,u,s} \geq 1 \tag{4.23}$$

$$u \in \{1, \ldots, (\tfrac{D}{7} - 1)\}, s \in \mathbf{S}$$

Af den sidste ligning ses, at mindst en af de binære variable skal være 1. Har den binære variabel en værdi på 1, tvinges det tilhørende udtryk til at evaluere til 2, hvilket netop svarer til en friweekend.

En fridag pr. 7 dage

Hver medarbejder skal have mindst en fridag pr. 7 løbende dage.

$$\sum_{d=i}^{i+6} x_{d,5,s} \geq 1, \;\; i = \{1, \ldots, D-6\}, s \in \mathbf{S} \tag{4.24}$$

Sovedagsrestriktioner

Man kan kun have sovedag dagen efter en nattevagt. I denne begrænsning udnytter vi, at $x_{d,t,s}$ er en binær variabel. Hvis man har sovedag dag $d+1$, så skal man have haft nattevagt dag d: $x_{d+1,4,s} = 1 \Rightarrow x_{d,3,s} = 1$, $s \in \{1, \ldots, S\}$, $d \in \{1, \ldots, D-1\}$. Dette bliver til:

$$x_{d,3,s} \geq x_{d+1,4,s}, \;\; d \in \mathbf{D}, s \in \mathbf{S} \tag{4.25}$$

Efter en nattevagt må der kun kunne tildeles enten en sovedag eller endnu en nattevagt. Det vil sige $x_{d,3,s} = 1 \Rightarrow x_{d+1,3,s} + x_{d+1,4,s} = 1, \;\; s \in \mathbf{S}, d \in \mathbf{D} \setminus \{D\}$.
Hvis man har nattevagt dag d, vil nedenstående udtryk give et ettal på venstresiden. Derved tvinges en af addenterne på højresiden til også at være en, hvilket er, hvad vi ønsker. Dermed er det sikret, at man enten har nattevagt eller sovedag efter en nattevagt.

$$x_{d,3,s} \leq x_{d+1,3,s} + x_{d+1,4,s}, \;\; d \in \mathbf{D} \setminus \{D\}, s \in \mathbf{S} \tag{4.26}$$

11-timersreglen

Da man ikke kan have mere end en vagttype pr. dag, er 11-timersreglen automatisk overholdt, hvis man har dagvagt. Det tidligste tidspunkt, man kan have en vagt på efter en dagvagt, er den efterfølgende dags dagvagt, hvilket vil sige 16 timer efter, man har fået fri.
Hvis man har nattevagt, er der allerede indlagt begrænsninger for, at man enten skal have en nattevagt eller en sovedag den efterfølgende dag, så her er der heller ingen problemer.

Det eneste sted, modellen specifikt skal have indlagt begrænsninger i forbindelse med 11-timersreglen, er efter aftenvagter. Hvis man har en aftenvagt dag d, vil der kun være 8 timer til dagvagten dag $d+1$ begynder. Denne kombination overholder ikke 11-timersreglen og skal derfor ikke være tilladt. Det vil sige $x_{d,2,s} = 1 \Rightarrow x_{d+1,1,s} = 0$.

Dette kan også udtrykkes ved, at summen af $x_{d+1,1,s} + x_{d,2,s}$ højst er 1, da en tildeling af den ene vagttype udelukker en tildeling af den anden.

$$x_{d,2,s} + x_{d+1,1,s} \leq 1, \;\; d \in \mathbf{D} \setminus \{D\}, s \in \mathbf{S} \tag{4.27}$$

Fordeling af nattevagter

De øvre og nedre grænser for hvor mange nattevagter en medarbejder må have, findes ved først at beregne andelen af nattevagter til hver sygeplejerske.

$$NDel_s = \frac{n_s \cdot h_s}{\sum_{i=1}^{S} n_i \cdot h_i}, \;\; s \in \mathbf{S} \tag{4.28}$$

Denne omregner vi til en øvre grænse for hvor mange nattevagter sygeplejerske s må have i alt.

$$NMax_s = \lceil NDel_s \cdot \sum_{d=1}^{D} b_{d,3} \rceil, \ \ s \in \mathbf{S} \tag{4.29}$$

Vi finder den nedre grænse for antallet af nattevagter ved at trække en fra $NMax_s$.

$$NMin_s = n_s \cdot (NMax_s - 1), \ \ s \in \mathbf{S} \tag{4.30}$$

Antallet af en sygeplejerskes nattevagter skal altså ligge inden for $NMin_s$ og $NMax_s$.

$$\sum_{d=1}^{D} x_{d,3,s} \geq NMin_s, \ s \in \mathbf{S} \tag{4.31}$$

$$\sum_{d=1}^{D} x_{d,3,s} \leq NMax_s, \ s \in \mathbf{S} \tag{4.32}$$

4.3 Bløde begrænsninger

I det følgende gennemgår vi de bløde begrænsninger, der *ønskes* overholdt.

Vagter deles ligeligt

Vi ønsker at finde bløde øvre og nedre grænser for antallet af vagter, man skal have. For at dele dag- og aftenvagter retfærdigt er det relevant at vide, hvor mange nattevagter sygeplejerske s har. Dette vides ikke med sikkerhed, før modellen er løst. Derfor går vi ud fra, at sygeplejerske s har $NDel_s$ ud af de $\sum_{d=1}^{D} b_{d,3}$, nattevagter. Vi definerer en nattevagt til at vare 9,75 timer og finder, hvor mange timer sygeplejerske s i alt har brugt på nattevagter. Dette trækkes fra ens samlede timetal på 4 uger. Det vil sige at det reviderede timetal, h'_s for sygeplejerske s bliver

$$h'_s = 4 \cdot h_s - \left(9,75 \cdot NDel_s \cdot \sum_{d=1}^{D} b_{d,3} \right), s \in \mathbf{S} \tag{4.33}$$

Vi finder andelen, $DADel_s$, af dag- og aftenvagter for sygeplejerske s ved at gå ud fra de reviderede timetal.

$$DADel_s = \frac{h'_s}{\sum_{i=1}^{S} h'_i}, \ s \in \mathbf{S} \tag{4.34}$$

Vi kan nu finde de bløde grænser for antallet af dag- og aftenvagter.

$$DMax_s = \lceil DADel_s \cdot \sum_{d=1}^{D} b_{d,1,s} \rceil, \ s \in \mathbf{S} \tag{4.35}$$

$$DMin_s = DMax_s - 1, \ s \in \mathbf{S} \tag{4.36}$$

$$AMax_s = \lceil DADel_s \cdot \sum_{d=1}^{D} b_{d,2,s} \rceil, \ s \in \mathbf{S} \tag{4.37}$$

$$AMin_s = AMax_s - 1, \ s \in \mathbf{S} \tag{4.38}$$

Disse bruges til at formulere begrænsningerne for, hvordan vagterne skal fordeles. Hvis en medarbejder har flere vagter end den øvre grænse, bidrager dette positivt til objektfunktionen. Hvis en medarbejder har færre vagter end den nedre grænse, bidrager det ligeledes positivt til objektfunktionen. Vi

indfører to variable for hver type vagt ($AMaxVar$, $AMinVar$ og $DMaxVar$ og $DMinVar$) for overtrædelserne. Variablene bidrager positivt i objektfunktionen. Først betragter vi dagvagterne

$$\sum_{d=1}^{D} x_{d,1,s} \leq DMax_s + DMaxVar_s,\ s \in \mathbf{S} \tag{4.39}$$

$$\sum_{d=1}^{D} x_{d,1,s} + DMinVar_s \geq DMin_s,\ s \in \mathbf{S} \tag{4.40}$$

Dernæst aftenvagterne

$$\sum_{d=1}^{D} x_{d,2,s} \leq AMax_s + AMaxVar_s,\ s \in \mathbf{S} \tag{4.41}$$

$$\sum_{d=1}^{D} x_{d,2,s} + AMinVar_s \geq AMin_s,\ s \in \mathbf{S} \tag{4.42}$$

Vi knytter nu prisen p_1 til antallet af overtrædelser og finder et bidrag, c_1 til objektfunktionen

$$p_1 \sum_{s=1}^{S} DMaxVar_s + DMinVar_s + AMaxVar_s + AMinVar_s = c_1 \tag{4.43}$$

Fridagsønsker

Det er muligt at specificere fridagsønsker for hver medarbejder. Til dette benyttes parametrene $o_{d,s} \in \mathbb{N}_0$. En værdi større end 0 svarer til, at sygeplejerske s ønsker fri dag d med prioriteten $o_{d,s}$. Hvis et fridagsønske ikke bliver opfyldt, vil værdien af $o_{d,s}$ indgå med en positiv værdi i objektfunktionen. Udover medarbejderens prioritering af ønsket er det også muligt at knytte en pris p_2 til denne omkostning. p_2 benyttes til at justere, hvor vigtigt det er for afdelingen at opfylde fridagsønsker i forhold til de andre bløde begrænsninger.

$$p_2 \cdot \sum_{s=1}^{S} \sum_{d=1}^{D} (o_{d,s} \cdot (1 - x_{d,5,s})) = c_2 \tag{4.44}$$

Som det ses, vil værdien af c_2 svare til omkostningen ved ikke at give personalet fri de dage, de ønsker det.

Weekendvagter

Vi er interesserede i at kalde så få medarbejdere som muligt på arbejde i weekenden. Dette klares ved, at vi for hver medarbejder forsøger at give denne samme vagttype lørdag og søndag.
Til dette introducerer vi de nye binære variable $bens_{t,u,s}$, der repræsenterer vagttype t i weekend u for sygeplejerske s.

$$x_{7 \cdot u-1,t,s} + x_{7 \cdot u,t,s} \geq 2 \cdot bens_{t,u,s},\ \ u \in \{1, \ldots, \tfrac{D}{7}\}, t \in \mathbf{T}, s \in \mathbf{S} \tag{4.45}$$

Hvis $bens_{t,u,s}$ er lig 1, vil sygeplejerske s have vagttype t både lørdag og søndag i weekend nummer u. Hvis $bens_{t,u,s}$ er lig 0, kan man ikke sige noget om sygeplejerskens vagttyper i den aktuelle weekend. Da vi helst vil have, at hver medarbejder har ens weekendvagter, ønsker vi at maksimere

summen af samtlige $bens_{t,u,s}$.
Vi summerer derfor over alle $bens_{t,u,s}$ og mulitplicerer med prisen p_3 for overtrædelsen.

$$p_3 \cdot \sum_{t=1}^{T} \sum_{u=1}^{\frac{D}{7}} \sum_{s=1}^{S} bens_{t,u,s} = ct_3 \tag{4.46}$$

Modsat de andre bløde begrænsninger er der her tale om en værdi, som vi ønsker at maksimere. Derfor skal variablen ct_3 indgå med negativ pris i objekfunktionen. Vi finder omkostningen c_3 til objektfunktionen.

$$0 - ct_3 = c_3 \tag{4.47}$$

Dette kan resultere i en negativ objektsfunktionsværdi. Hvis hver medarbejder har ens vagttyper i alle weekender giver dette en negativ værdi på $S * \frac{D}{7}$, denne værdi kan man blot lægge til i objektfunktionen, hvis man ønsker en objektfunktion med mindsteværdi på nul.

Vi har et ønske om, at en sovevagt lørdag kombineret med en fridag søndag skal være gratis. Under den endelige afprøvning af GAMS-implementeringen viste det sig dog, at dette forhold gav os uforudsete problemer. Vi har derfor besluttet ikke at medtage forholdet i den matematiske model.
Dette betyder at den matematiske model nu har færre muligheder for at lægge weekendvagter, der ikke koster noget i objektfunktionen.

Vagtrækkefølge ([sove,nat])

Modellen skal forsøge at undgå, at en medarbejder får tildelt en nattevagt dagen efter en sovedag. Til denne begrænsning introducerer vi endnu en binær variabel $bv_{d,s}$, som vil være lig 1, hvis en medarbejder har en sovedag dag d og en nattevagt dag $d+1$.

$$x_{d,4,s} + x_{d+1,3,s} - 1 \leq bv_{d,s}, \;\; d \in \mathbf{D} \setminus \{D\}, s \in \mathbf{S} \tag{4.48}$$

Til brug i objektfunktionen finder vi summen af samtlige $bv_{d,s}$ og ganger med prisen p_4 for overtrædelsen.

$$p_4 \cdot \sum_{d=1}^{D-1} \sum_{s=1}^{S} bv_{d,s} = c_4 \tag{4.49}$$

Umiddelbart kan $bv_{d,s}$ altid antage værdien 1, da begrænsningen stadig vil være opfyldt. Eftersom objektfunktionen netop skal minimere summen af samtlige $bv_{d,s}$, vil modellen forsøge at få alle $bv_{d,s}$ til at være lig 0.

Objektfunktion

Den endelige objektfunktion skal minimere priserne på de bløde begrænsinger.

$$\text{Min} \sum_{i=1}^{4} c_i \tag{4.50}$$

4.4 Endelig model

Den samlede formulering af den matematisk model bliver derved til:

$$\text{Min} \sum_{i=1}^{4} c_i \tag{4.50}$$

uht.

$$\sum_{s=1}^{S} x_{d,a,s} = b_{d,a}, \;\; d \in \mathbf{D}, a \in \mathbf{A} \tag{4.1}$$

$$\sum_{t=1}^{T} x_{d,t,s} = 1, \;\; d \in \mathbf{D}, s \in \mathbf{S} \tag{4.2}$$

$$x_{d,3,s} \leq n_s, \;\; s \in \{1, \ldots, S\}, d \in \{1, \ldots, D\} \tag{4.3}$$

$$x_{d,5,s} + x_{d+1,5,s} \geq 2 \cdot ff_{d,s}, \;\; d \in \mathbf{D} \setminus \{D\}, s \in \mathbf{S} \tag{4.4}$$

$$x_{d,1,s} + x_{d+1,5,s} + x_{d+2,3,s} \geq 3 \cdot dfn_{d,s}, \;\; d \in \mathbf{D} \setminus \{D\}.s \in \mathbf{S} \tag{4.5}$$

$$x_{d,1,s} + x_{d+1,5,s} + x_{d+2,1,s} \geq 3 \cdot dfd_{d,s}, \; d \in \mathbf{D} \setminus \{D\}, s \in \mathbf{S} \tag{4.6}$$

$$x_{d,1,s} + x_{d+1,5,s} + x_{d+2,2,s} \geq 3 \cdot dfa_{d,s}, \; d \in \mathbf{D} \setminus \{D\}, s \in \mathbf{S} \tag{4.7}$$

$$x_{d,2,s} + x_{d+1,5,s} + x_{d+2,2,s} \geq 3 \cdot afa_{d,s}, \; d \in \mathbf{D} \setminus \{D\}, s \in \mathbf{S} \tag{4.8}$$

$$x_{d,2,s} + x_{d+1,5,s} + x_{d+2,3,s} \geq 3 \cdot afn_{d,s}, \; d \in \mathbf{D} \setminus \{D\}, s \in \mathbf{S} \tag{4.9}$$

$$x_{d,4,s} + x_{d+1,5,s} + x_{d+2,2,s} \geq 3 \cdot sfa_{d,s}, \; d \in \mathbf{D} \setminus \{D\}, s \in \mathbf{S} \tag{4.10}$$

$$x_{d,4,s} + x_{d+1,5,s} + x_{d+2,3,s} \geq 3 \cdot sfn_{d,s}, \; d \in \mathbf{D} \setminus \{D\}, s \in \mathbf{S} \tag{4.11}$$

$$u2ff_s = \sum_{d=8}^{13} ff_{d,s}, \;\; s \in \mathbf{S} \tag{4.12}$$

$$u2dfn_s = \sum_{d=7}^{13} dfn_{d,s}, \;\; s \in \mathbf{S} \tag{4.13}$$

$$u2dfd_s = \sum_{d=7}^{13} dfd_{d,s}, \;\; s \in \mathbf{S} \tag{4.14}$$

$$u2dfa_s = \sum_{d=7}^{13} dfa_{d,s}, \;\; s \in \mathbf{S} \tag{4.15}$$

$$u2afa_s = \sum_{d=7}^{13} afa_{d,s}, \;\; s \in \mathbf{S} \tag{4.16}$$

$$u2afn_s = \sum_{d=7}^{13} afn_{d,s}, \;\; s \in \mathbf{S} \tag{4.17}$$

$$u2sfa_s = \sum_{d=7}^{13} sfa_{d,s}, \;\; s \in \mathbf{S} \tag{4.18}$$

$$u2sfn_s = \sum_{d=7}^{13} sfn_{d,s}, \;\; s \in \mathbf{S} \tag{4.19}$$

$$2 \cdot u2ff_s + 2 \cdot u2dfn_s + u2dfd_s + u2dfa_s + u2afa_s + u2afn_s + u2sfa_s + u2sfn_s \geq 2 \tag{4.20}$$

$$x_{7 \cdot u-1,5,s} + x_{7 \cdot u,5,s} \geq 2 \cdot bweek_{1,u,s}, \; u \in \{1, \ldots, (\tfrac{D}{7} - 1)\}, s \in \mathbf{S} \tag{4.21}$$

$$x_{7 \cdot u+6,5,s} + x_{7 \cdot u+7,5,s} \geq 2 \cdot bweek_{2,u,s}, \; u \in \{1, \ldots, (\tfrac{D}{7} - 1)\}, s \in \mathbf{S} \tag{4.22}$$

$$bweek_{1,u,s} + bweek_{2,u,s} \geq 1, \; u \in \{1, \ldots, (\tfrac{D}{7} - 1)\}, s \in \mathbf{S} \tag{4.23}$$

$$\sum_{d=i}^{i+6} x_{d,5,s} \geq 1, \; i = \{1, \ldots, D-6\}, s \in \mathbf{S} \tag{4.24}$$

$$x_{d,3,s} \geq x_{d+1,4,s}, \; d \in \mathbf{D}, s \in \mathbf{S} \tag{4.25}$$

$$x_{d,3,s} \leq x_{d+1,3,s} + x_{d+1,4,s}, \; d \in \mathbf{D} \setminus \{D\}, s \in \mathbf{S} \tag{4.26}$$

$$x_{d,2,s} + x_{d+1,1,s} \leq 1, \; d \in \mathbf{D} \setminus \{D\}, s \in \mathbf{S} \tag{4.27}$$

$$NDel_s = \frac{n_s \cdot h_s}{\sum_{i=1}^{S} n_i \cdot h_i},\ s \in \mathbf{S} \tag{4.28}$$

$$NMax_s = \lceil NDel_s \cdot \sum_{d=1}^{D} b_{d,3} \rceil,\ s \in \mathbf{S} \tag{4.29}$$

$$NMin_s = n_s \cdot (NMax_s - 1),\ s \in \mathbf{S} \tag{4.30}$$

$$\sum_{d=1}^{D} x_{d,3,s} \geq NMin_s,\ s \in \mathbf{S} \tag{4.31}$$

$$\sum_{d=1}^{D} x_{d,3,s} \leq NMax_s,\ s \in \mathbf{S} \tag{4.32}$$

$$h'_s = 4 \cdot h_s - \left(9,75 \cdot NDel_s \cdot \sum_{d=1}^{D} b_{d,3}\right), s \in \mathbf{S} \tag{4.33}$$

$$DADel_s = \frac{h'_s}{\sum_{i=1}^{S} h'_i},\ s \in \mathbf{S} \tag{4.34}$$

$$DMax_s = \lceil DADel_s \cdot \sum_{d=1}^{D} b_{d,2,s} \rceil,\ s \in Sm \tag{4.35}$$

$$DMin_s = DMax_s - 1,\ s \in Sm \tag{4.36}$$

$$AMax_s = \lceil DADel_s \cdot \sum_{d=1}^{D} b_{d,1,s} \rceil,\ s \in Sm \tag{4.37}$$

$$AMin_s = AMax_s - 1,\ s \in Sm \tag{4.38}$$

$$\sum_{d=1}^{D} x_{d,1,s} \leq DMax_s + DMaxVar_s,\ s \in \mathbf{S} \tag{4.39}$$

$$\sum_{d=1}^{D} x_{d,1,s} + DMinVar_s \geq DMin_s,\ s \in \mathbf{S} \tag{4.40}$$

$$\sum_{d=1}^{D} x_{d,2,s} \leq AMax_s + AMaxVar_s,\ s \in \mathbf{S} \tag{4.41}$$

$$\sum_{d=1}^{D} x_{d,1,s} + AMinVar_s \geq AMin_s,\ s \in \mathbf{S} \tag{4.42}$$

$$p_1 \sum_{s=1}^{S} DMaxVar_s + DMinVar_s +$$
$$AMaxVar_s + AMinVar_s = c_1 \tag{4.43}$$

$$p_2 \cdot \sum_{s=1}^{S} \sum_{d=1}^{D} \left(o_{d,s} \cdot \left(1 - x_{d,5,s}\right)\right) = c_2 \tag{4.44}$$

$$x_{7 \cdot u-1,t,s} + x_{7 \cdot u,t,s} \geq 2 \cdot bens_{t,u,s},\ u \in \{1, \ldots, \tfrac{D}{7}\}, t \in \mathbf{T}, s \in \mathbf{S} \tag{4.45}$$

$$p_3 \cdot \sum_{t=1}^{T} \sum_{d=1}^{\frac{D}{7}} \sum_{s=1}^{S} bens_{t,d,s} = ct_3 \tag{4.46}$$

$$0 - ct_3 = c_3 \tag{4.47}$$

$$x_{d,4,s} + x_{d+1,3,s} - 1 \leq bv_{d,s},\ d \in \mathbf{D} \setminus \{D\}, s \in \mathbf{S} \tag{4.48}$$

$$p_4 \cdot \sum_{d=1}^{D-1} \sum_{s=1}^{S} bv_{d,s} = c_4 \tag{4.49}$$

Ligning 4.12 til 4.20 er alle specifikke for uge nummer to. Lignende ligninger for uge nummer et, tre og fire skal medtages, hvis modellen skal være komplet.

Som man kan se, bliver modellen hurtigt meget stor En ekstra begrænsning eller sygeplejerske tilføjer mange ekstra variable. Dette gør modellen meget tung at arbejde med. Desuden bliver den samlede

model hurtigt meget uoverskuelig. Ydermere bidrager de mange binære variable, der repræsenterer diskrete valg i løsningsrummet, en del til kompleksiteten.

Vi har implementeret ovenstående model i det matematiske modelleringssprog GAMS. GAMS står for "General Algebraic Modelling System" og har flere forskellige solvere indbygget. Her har vi benyttet "MIP"-solveren, da det er et blandet heltalsproblem, vi har med at gøre. Kildekoden kan ses i bilag C.2.

Formålet med denne rapport er primært at arbejde med CP, så GAMS-modellen tjener blot det formål at sammenligne CP med et traditionelt lineært programmeringssprog, hvilket er en af årsagerne til, at vi flere steder har valgt at ignorere mulighederne for en mere fleksibel model.

Kapitel 5

Constraint programming

I dette kapitel giver vi et indblik i, hvad constraint programming er. Vi beskriver virkemåden, og endvidere belyser vi emnet ved hjælp af små eksempler indført i CP-sproget ECL^{i}PS^e .

5.1 Baggrund

CP er et paradigme rettet mod effektiv løsning af kombinatoriske problemer. Det er opbygget af teknikker fra matematik, kunstig intelligens og operationsanalyse.

Efterhånden er CP blevet et praktisk, anvendeligt værktøj til kendte problemstillinger såsom "jobshop-scheduling" [2], "warehouse location" [12] og vagtplanlægning [13], [6].

Metoden har vist sig at være effektiv både med hensyn til hurtig programudvikling og med hensyn til modellering af vanskelige problemer, idet den tilbyder helt andre fremgangsmåder end de gængse programmeringssprog. I dag benytter blandt andet [3] British Airways, SAS, de franske jernbaner (SNCF) og Michelin CP i udviklingen.

Historie

De første moderne CP-sprog var udvidelser af logikprogrammeringssprog. Disse kaldes derfor også for *constraint logic programming*-sprog (CLP). De opstod ved udvidelse af eksisterende logikprogrammeringssprog (for eksempel Prolog) ved at erstatte daværende metoder med tests for "constraint satisfaction". Disse CP-sprog udsprang fra forskning i Europa og Australien i slutningen af 80'erne.

I løbet af 90'erne blev interessen for CP fornyet. Den fornyede interesse skyldtes, at de nye programmeringssprog var bedre og åbnede flere muligheder end de gamle, samt at der var kommet bedre og mere kraftfulde solvere.
CP gør det muligt at beskrive problemerne på et højt niveau, hvilket igen gør problemerne lettere at udtrykke.

Deklarativ metode

CP er et deklarativt sprog. Programmering i CP foregår ved, at man definerer et problem ved hjælp af variable og begrænsninger. Begrænsningerne (på engelsk kaldet constraints) gælder for en eller flere af variablene og forsøges opfyldt, når systemet leder efter en løsning til problemet. Afhængigt af hvilket problem der ønskes løst, og hvilken slags løsning man specificerer, vil systemet finde enten en mulig løsning, en god løsning, den bedste løsning, eller alle løsninger, hvis en løsning eksisterer.

Groft sagt er fremgangsmåden den, at man "blot" definerer problemet, som derefter bliver løst af run-time-systemet. Denne deklarative tilgang til problemløsningen er anderledes end den sædvanlige algoritmiske. I imperative programmeringssprog, såsom `C`, `C++` og `Pascal`, definerer man skridt for skridt, hvad der skal udføres for at finde en løsning. Hvis man ønsker at komme fra et udtryk X til et andet udtryk Y, skal man direkte definere, *hvordan* man kommer fra X til Y.
I deklarative programmeringssprog bliver programmerne bygget op af definitioner og udsagn, som i stedet beskriver forholdene i det problem, man ønsker løst. Hvis man her ønsker at komme fra X til Y, beskriver man i stedet *forholdet* mellem de to udtryk. I afsnit 5.2.2, vises vi eksempler på hvordan dette ser ud i CP.

Det er ofte nyttigt at vurdere, hvordan man beskriver sit problem på den bedste måde. I praksis kan man vinde meget ved at opbygge sit problem hensigtsmæssigt, da søgningen efter en brugbar løsning herved kan lettes markant. Dette vil vi komme nærmere ind på i afsnit 5.3. I afsnit 5.4 kommer vi kort ind på søgestrategier, og i afsnit 5.5 giver vi et eksempel på CP brugt i praksis.

Fleksibilitet

Når man forsøger at løse komplicerede problemer i almindelige programmeringssprog, benytter man ofte en af to traditionelle fremgangsmåder. Enten laver programmøren selv en algoritme til at løse problemet med, eller også benyttes der en generel model, som passer til den klasse, det aktuelle problem hører ind under.

Den første fremgangsmåde er ofte dyr og besværlig, da det tager tid at bygge en fyldestgørende algoritme op fra bunden. Desuden vil den i sidste ende være meget svær at tilpasse, hvis den senere skal bruges til et andet problem, eller hvis det oprindelige problem ændres en smule. Den anden metode bliver meget let for generel og vil sandsyndligvis ikke være så detaljeret, at alle begrænsningerne kan udtrykkes naturligt, hvis de i det hele taget kan udtrykkes. Derudover kan der også være problemer med fleksibiliteten i forbindelse med at benytte forskellige specifikke heuristikker til løsning af problemet.

De moderne CP-sprog åbner muligheden for at udvikle fleksible programmer. De tilbyder et programmeringssprog bygget ovenpå en *"constraint solver"*. Dette betyder, at programmøren meget let kan definere sit problem og indkode forskellige domænespecifikke heuristikker, mens der stadig ligger generelle løsningsteknikker i run-time-systemet.

5.2 Hvordan virker constraint programming

Hele ideen bag CP er at løse problemer ved at definere begrænsninger i form af egenskaber og betingelser. En løsning til problemet er kendetegnet ved, at alle begrænsninger er overholdt. Jo flere begrænsinger der indføres, jo mere mindskes løsningsrummet, og jo hurtigere kan løsningen findes. På denne måde bruges begrænsninger både til at beskære løsningsrum med og til at kontrollere, om en given løsning er mulig.

En af styrkerne i CP er, at det er let at definere begrænsninger for og mellem variable, selv hvis begrænsningerne er meget komplekse. Dette gør, at CP er en velegnet metode til løsning af svære kombinatoriske problemer. Disse problemer kan dels være vanskelige at udtrykke i de imperative programmeringssprog og dels være temmelig svære at finde en løsning til inden for rimelig tid.

Grunden til, at disse problemer kan være svære at udtrykke i imperative sprog, er, at de ofte indeholder komplicerede begrænsninger.
I stedet for relativt simple begrænsninger som for eksempel

$$x \geq 4 \ , \tag{5.1}$$

kan der forekomme begrænsninger af variable, som afhænger af andre variable

$$x - y > 2 \tag{5.2}$$

og der kan være begrænsninger, som deler variablenes domæner op i flere intervaller

$$x > 4 \ \vee \ x < -1 \tag{5.3}$$

I CP er alle disse typer af begrænsninger lige lette at udtrykke.

En væsentlig grund til, at kombinatoriske problemer kan være svære at løse, er, at løsningsrummene kan vokse eksponentielt med antallet af variable, samt at de ikke nødvendigvis er konvekse. Desuden er mange kombinatoriske problemer NP-komplette, hvilket betyder, at det sjældent kan lade sig gøre at søge hele løsningsrummet igennem for at finde den bedste løsning. Dette betyder at det uanset løsningsværktøj, er nødvendigt at benytte en specifik søgerutine til at finde en god løsning.

5.2.1 Constraint satisfaction problems

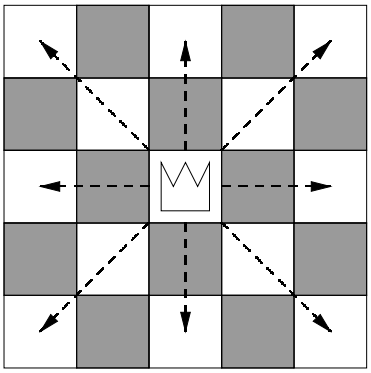
Langt de fleste problemer, man beskæftiger sig med i forbindelse med operationsanalyse, er af typen *constraint satisfaction problems* (CSP).

CSP er defineret ved, at der haves en mængde af *variable*, som alle er bundet til hver deres *domæne*, samt en mængde af *begrænsninger* mellem en eller flere variable.

En løsning til et CSP er en tildeling af værdier til alle variable på en sådan måde, at værdierne er indeholdt i variablenes domæner, og så aller begrænsningerne er overholdt. Afhængig af problemet ønsker man enten at finde blot en mulig løsning, alle løsninger eller den bedste løsning.

Et eksempel på et CSP er det kendte N-dronningeproblem. Problemet går ud på at placere N dronninger, på et skakbræt med $N \times N$ felter, uden at dronningerne kan slå hinanden.

På figur 5.1 ses det, at en brik kommer til at spærre i 8 retninger ud fra sig (medmindre den står helt ud til en af kanterne).



Figur 5.1: En dronning dækker mange retninger

Løsningen til problemet er at placere de N dronninger således, at to dronninger ikke står i samme række eller søjle, endvidere må de heller ikke stå på samme skrå linje parallel med diagonalen, hverken i den ene eller den anden retning.

Problemet består af nogle variable med hver deres domæne. Variablene beskriver hvilke felter dronningerne står på. Endvidere vil der være begrænsninger, der beskriver hvorledes dronningerne kan (eller ikke kan) stå i forhold til hinanden. I afsnit 5.5.1 viser vi, hvordan dette problem kan løses ved hjælp af CP.

Løsningen til et CSP kan enten findes ved systematisk søgning af hele løsningsrummet, eller man kan benytte en søgemekanisme der kun ser på dele af løsningsrummet.

5.2.2 Relationer

I CP angives begrænsninger i et problem, som relationer mellem variable. Ud fra en relation kan man:

- finde gyldige værdier for en variabel, givet værdier for de andre variable
- undersøge om en bestemt tildeling af værdier til de variable er lovlig
- finde et forhold mellem to eller flere variable

En relation er også et prædikat, og vi vil bruge begge betegnelser fremover alt efter, hvad der passer bedst ind i sammenhængen.

Man definerer altså ikke funktioner som $f(x) = y$, hvor man kun kan finde y ud fra x. Man definerer i stedet en *relation* mellem y og x, hvorved det også er muligt at finde x givet en værdi for y. Derudover kan man finde ud af, om to givne værdier af x og y er lovlige i henhold til den definerede relation. I sidste tilfælde vil resultatet være en sandhedsværdi. Hvis der er flere variable i en begrænsning, kan man endvidere se, hvordan disse afhænger af hinanden og undersøge hvilke konsekvenser, det har for nogle variable, hvis man ændrer på andre variable.

I de følgende eksempler viser vi, hvordan man definerer og arbejder med relationer i CP. Det første eksempel er et program[1], der er i stand til at regne med Newtons 2. lov ($F_{res} = m \cdot a$).

```
newtons_2(F,M,A) :-
                F #= M*A.
```

Programmet beskriver forholdet (relationen) mellem de tre variable. Ved hjælp af denne relativt simple kode er det muligt at bestemme værdien af den resulterende kraft (F_{res}), massen (m) eller accelerationen (a) ud fra de to andre variable.
Dette skal ses i modsætning til et program skrevet i for eksempel `C`, hvor en funktion umiddelbart kun vil være i stand til at finde netop én af de tre variable.

Kender man massen ($m = 2$) og accelerationen ($a = 3$), vil man med følgende kommando kunne finde den resulterende kraft (F_{res}):

```
[eclipse ]: newtons_2(F,2,3).

F = 6
```

Hvis man i stedet kender den resulterende kraft ($F_{res} = 6$) og massen ($m = 2$) og ønsker at finde accelerationen (a), kan man stadig bruge det samme program:

```
[eclipse ]: newtons_2(6,2,A).

A = 3
```

Programmet kan også bruges til at tjekke om et givet resultat er gyldigt:

```
[eclipse ]: newtons_2(6,2,3).
Yes
[eclipse ]: newtons_2(6,2,7).
No
```

[1] Eksemplet her er inspireret af et lignende eksempel fra [12]

Som det ses, svarer programmet tilbage med "`yes`"eller "`no`", for at indikere at den angivne løsning *er*, hhv. *ikke er* en gyldig løsning på problemet.

Det er ikke nødvendigt at udføre en beregning eller sammenligning i et prædikat. Hvis man vil angive en tilstand eller information, såsom at en variabel for eksempel "solen" er gul, kan man oprette et prædikat gul og angive følgende

```
gul(sol).
```

Dette bevirker, at hver gang gul bliver kaldt med et argument, forsøger solveren at sætte argumentet til sol. Er dette muligt, evaluerer udtrykket til sandt og ellers til falskt.

Ønsker man at udtrykke, at flere forhold skal være opfyldt samtidig, er dette også muligt.

```
minusen(X,Y) :- X #> 0, Y #= X-1.
```

Ovenstående udtrykker, at `X` skal være større end nul, *og* at `Y=X-1`.

En relation behøver ikke kun at have én evalueringsmulighed. Betragter vi for eksempel fakultetsfunktionen, er der to interessante tilfælde, et hvor argumentet er 0 og et, hvor det er større end 0. Dette kan løses ved hjælp af en sætning, hvis man bruger et `if-then-else`-udtryk, men det er mere overskueligt med to udtryk:

```
fac(0,1).
fac(N,Y) :- N #>0, NT is N-1,
            fac(NT,X), Y is N*X.
```

Når solveren behandler et kald af `fac`, vil den først forsøge at matche kaldet med `fac(0,1)`, kan dette ikke lade sig gøre, vil den dernæst forsøge at udføre `fac(N,Y)`. `fac(N,Y)` kræver, at `N` er større end 0, og kalder dernæst `fac(N-1,Y)`. Grunden til, at der her er brugt et `is`-udtryk, er, at det med det samme omregner værdien af variablen til tal. Almindeligt `#=` gemmer blot udtrykket `N-1` uden at evaluere det, og det går derfor galt, når man kalder funktionen med et udtryk i stedet for en værdi. Udtrykket 1-1 matcher for eksempel ikke til nul men blot til udtrykket X-Y.

5.2.3 Constraint propagation og domænereduktion

Den grundlæggende teknik til problemløsningen i CP er *constraint propagation*. Constraint propagation er igen en term, der ikke direkte har et dansk ord. De bedste bud er videreformidling eller udbredelse. Constraint propagation går ud på at se på sammenhængen mellem variable i relationer og deres domæner. Hvis to variable afhænger af hinanden (står i relation til hinanden), videreformidles viden om den ene variabels domæne til den anden variabel. Denne viden kan dernæst bruges til at foretage en domænereduktion.

Som eksempel kan vi se på to variable x og y med relationen

$$y = 2 \cdot x$$

Vi fastsætter domænerne for x og y til

$$D_x = \{1, 2, \ldots, 10\}$$
$$D_y = \{1, 2, \ldots, 10\}$$

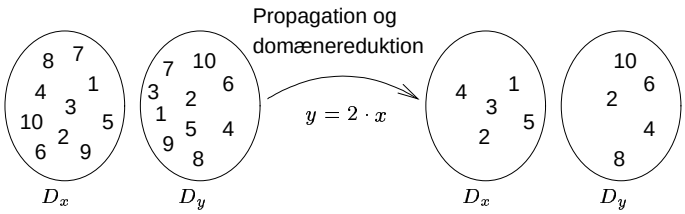
Da y er lig det dobbelte af x, og D_x udelukkende består af heltal, må y nødvendigvis være et lige tal. Dette medfører, at domænet for y reduceres til:

$$D_y = \{2, 4, 6, 8, 10\}$$

Da y ikke kan være større end 10, kan x ikke være større end $\frac{10}{2} = 5$, hvilket medfører, at D_x reduceres til

$$D_x = \{1, 2, 3, 4, 5\}$$

På denne måde foretages domænereduktioner for alle de variabeldomæner, hvor de variable relaterer til hinanden. I figur 5.2 har vi illustreret resultatet af den beskrevne domænereduktion.



Figur 5.2: Domænereduktion i CP

5.2.4 Cykler

Der kan opstå problemer, hvis variablene er indbyrdes afhængige af hinanden på en sådan måde, at der opstår cykler. Betragter vi følgende eksempel

```
cyk(Y) :- X1 is Y+1, cyk1(X1).
cyk1(Y) :- X1 is Y-1, cyk(X1).
```

vil et kald som `cyk(4)` ikke terminere men blot gentagent kalde `cyk1(5)` og `cyk(4)`. I dette eksempel er det naturligvis let at gennemskue, at der er en cykel, men i mere komplicerede programmer kan det godt være svært at holde styr på, hvornår der er cykler.

Hvis det ikke handler om ændring af variable men om domænereduktioner ville programmet ikke køre i ring. Når først et domæne er indskrænket til for eksempel $[3, \ldots, 5]$ kan det ikke blive ændret til noget større som eksempelvis $[4, \ldots, 7]$, da den begrænsning, der udtrykker det første domæne, stadig skal gælde.

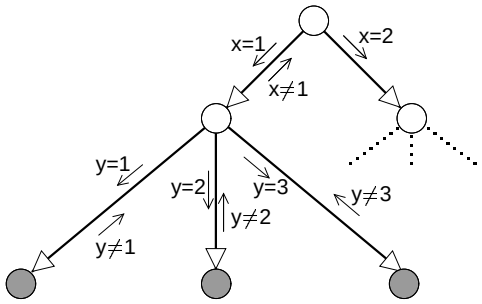
Constraint propagation og domænereduktion kan give nogen information om løsningen til et CSP, men finder ikke nødvendigvis en løsning. Hvis begrænsningerne ikke er stærke nok, vil de blot bevirke, at løsningsrummet mindskes. For derefter at finde en god eller en optimal løsning er det nødvendigt at benytte en søgestrategi.

5.3 Opbygning af program

Vi har nu set, hvordan man kan modellere problemer ved blot at definere hvilke sammenhænge, der gælder. Det er imidlertid ikke altid tilfældet, at en sådan definition er udtryksfuld nok, til at den indbyggede solver kan løse problemet effektivt. Det kan ske, at det tager for lang tid at finde en løsning eller i værste fald, at programmet slet ikke finder en løsning. For at kunne undgå dette, er det nødvendigt at forstå, hvordan den indbyggede solver virker.

Når programmet udføres, er det den underliggende tekniks mål at finde en lovlig tildeling af værdier til de præsenterede variable, således at alle reglerne er opfyldt. Man siger, at systemet evaluerer målet.
Målet evalueres ved at opbygge et søgetræ og gennemsøge dette ved hjælp af dybde-først-søgning fra venstre mod højre, som illustreret i figur 5.3. Dette kaldes *pre-order*-søgning. I figuren repræsenterer

hver knude et valg, der skal foretages, og hvert blad en løsning. Er løsningen ikke mulig, søges tilbage til det sidst foretagede valg og et nyt valg forsøges. Denne metode kaldes *backtracking*. Er der ikke flere valg at foretage, går man endnu et skridt tilbage og foretager et nyt valg. Således fortsættes, indtil hele træet er udforsket, eller en gyldig løsning er fundet.



Figur 5.3: Dybdeførst-søgning

Ønsker man mere end én løsning, gemmes den fundne løsning, og der søges videre, indtil hele træet er gennemsøgt. En sådan søgning kaldes en *komplet søgning*. Da træstørrelsen vokser eksponentielt med antallet af variable, og da antallet af variable kan være meget stort, vil en komplet søgning tit være meget tidskrævende.

Det er for dyrt både med hensyn til hukommelse og med hensyn til tidsforbrug at opbygge hele træet fra starten af, og derfor opbygges søgetræet undervejs i søgningen. Dette er især tidsbesparende, hvis man bare ønsker én løsning, og hvis denne findes i venstre del af træet, eller hvis store dele af træet kan skæres fra undervejs.

Oftest er det gennemsøgningen af træet, der tager tid. Derfor er det fornuftigt at tænke over, hvordan man bygger sit problem op, både for at få et effektivt program men også for at sikre at det terminerer. I følgende afsnit har vi et eksempel på ikke-termination og kommer ind på nogle af de ting, der har betydning for gennemsøgningen.

5.3.1 Regelrækkefølge

Når træet gennemsøges, bliver de regler, der står først i programmet, altid testet først. Dette har stor betydning for programkørslen.

Som eksempel betragter vi et program[2], der skal kunne beregne summen af tallene fra 1 til N for et givet N. Programmet kan bestå af følgende to regler.

$$\mathtt{sum}(N, S+N) :- \mathtt{sum}(N-1, S). \quad \text{(R1)}$$

$$\mathtt{sum}(0, 0). \quad \text{(R2)}$$

Dette går dog hurtigt galt. Ønsker man for eksempel at finde sum(1,S), vil solveren for hvert gennemløb starte med at se på den første regel (R1) og benytte denne, da den er generel nok til at passe til enhver inddata:

$$\mathtt{sum}(1, S) \rightarrow \mathtt{sum}(0, S') \rightarrow \mathtt{sum}(-1, S'') \rightarrow \mathtt{sum}(0-2, S''') \ldots$$

Dette kald vil aldrig terminere, da solveren aldrig vil komme ud af det første kald, men blot fortsætte med at teste for R1.

[2]Dette eksempel er lånt fra [12] side 217.

Dette kan man rette op på ved at bytte om på rækkefølgen af reglerne, så den mest generelle regel er til sidst, og specialtilfældet er først:

$$\mathsf{sum}(0,0). \tag{R2}$$
$$\mathsf{sum}(N, S+N) :- \mathsf{sum}(N-1, S). \tag{R1}$$

Nu vil solveren på det før nævnte input først teste udtrykket op mod R2, og terminere i andet gennemløb og give resultatet 1:

$$\mathsf{sum}(1,S) \rightarrow \mathsf{sum}(0,S') \rightarrow \mathsf{sum}(0,0)\ .$$

5.3.2 Udtømmende regler

Det er imidlertid ikke nok, at rækkefølgen af reglerne er fornuftig. Reglerne skal også være udtømmende. Prøver vi at kalde sum(1,0), vil programmet ikke komme ind i regel R2, da der ikke står nul på begge pladser samtidig, derfor vil R1 blive benyttet:

$$\mathsf{sum}(1,0) \rightarrow \mathsf{sum}(0,-1) \rightarrow \mathsf{sum}(-1,-1), \rightarrow \mathsf{sum}(-2,0)\ldots$$

Vi tilføjer begrænsningen $N \geq 1$ i R1 og får i stedet R3:

$$\mathsf{sum}(0,0). \tag{R2}$$
$$\mathsf{sum}(N, S+N) :- N >= 1, \mathsf{sum}(N-1, S). \tag{R3}$$

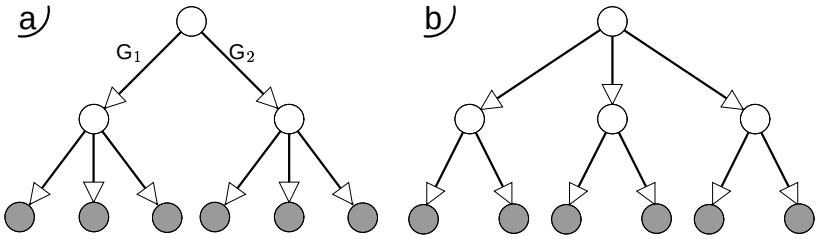
Nu vil programmet terminere og svare no, da summen af 1 ikke er 0.

5.3.3 Ordning af literaler

Som vi har set, kan en enkelt regel godt bestå af flere dele. Hver del kaldes en literal. Reglen R3 indeholder for eksempel 2 literaler, en der udtrykker, at N skal være større end 1, og en der udtrykker, hvad der egenlig skal gøres. Ligesom det er tilfældet med regler, testes literalerne i en regel i den rækkefølge, de står i. Havde $N \geq 1$ stået sidst, havde systemet kaldt det rekursive kald hele tiden og var aldrig nået til at teste om $N \geq 1$. Ordning af literaler er især vigtig, når reglen indeholder rekursive kald eller kalder andre regler.

5.3.4 Søgetræets størrelse

Vi har nu set, at rækkefølgen *af* reglerne er vigtig, og at rækkefølgen *i* reglerne er vigtig. Derudover har søgetræets størrelse også meget at sige. Ikke alene antallet af variable har indflydelse, men også rækkefølgen de behandles i. Generelt vil man gerne have et så lille søgetræ som muligt. Antallet af blade og antallet af niveauer i søgetræet vil i sidste ende være det samme uanset opbygningen, eftersom antallet af variable og deres kombinationer ikke afhænger af rækkefølgen. Alligevel kan der være stor forskel på antallet af valg, såkaldte *choicepoints*, der skal foretages undervejs i søgningen. Et *choicepoint* er en knude med to eller flere grene ud fra sig, det vil sige en knude, hvor et valg skal foretages.



Figur 5.4: Søgetræers størrelse

I figur 5.4 kan man se, at det er bedst at foretage valg for variable med små domæner først. Står man i en knude og skal vælge mellem gren G_1 og gren G_2, vil de to træer under grenene se ens ud, bortset fra det træerne evaluerer til. Da man gentager det resterende søgetræ for hver gren, og det er en fordel at gentage det så få gange som muligt, vil det være mest hensigtsmæssigt at have de valg med færrest muligheder først. Dette ses også på figuren, hvor der er færre choicepoints og større underliggende træer i figur a). Større underliggende træer er en fordel, ligeså snart man kan skære undertræer væk. Hvis man skærer hele venstre gren på søgetræet væk, vil reduktionen være større i figur 5.4 a) end i figur 5.4 b).

5.3.5 Effektivitet

En sidste ting der kan være med til at forbedre søgningen, er at lægge de valg, man forventer vil være mest succesfulde, længst til venstre i træet. På denne måde er der flest løsninger i den venstre del af træet, der jo bliver gennemsøgt først. Ønsker man at kende alle mulige løsninger, er denne fordeling overflødig, da man alligevel skal igennem hele træet.

En af ulemperne ved denne metode er, at den er meget problemspecifik. Derudover kan det være svært at bedømme, hvilke valg der går godt, og hvilke der går skidt. Som regel er det nødvendigt at afprøve implementeringen af denne metode empirisk.

5.3.6 Beskæring af søgetræ

Beskæring går ud på at skære dele af søgetræet væk. Jo højere oppe i træet beskæringen foregår, jo større del skæres fra. Generelt er det mest effektivt at gå frem efter *"failure-first"*-princippet, hvilket vil sige, at de valg, der mest sandsynligt fører til fejl, skal foretages i toppen af træet. Bemærk at vi her snakker om toppen af træet og ikke toppen *og* venstresiden af træet. Det kan være svært at bedømme, hvilke valg der med størst sandsynlighed fører til fejl. Her er man igen nødt til at prøve sig frem.

Nedenfor omtaler vi nogle af de muligheder, man har for at beskære søgetræet.

Overflødig begrænsning

Nogle gange kan det være hensigtsmæssigt at tilføje en overflødig begrænsning. Begrænsningen ændrer ikke på løsningsrummet, men gør det muligt at opdage ugyldige tildelinger tidligere. På denne måde kan søgetræet beskæres tidligere end ellers. Jo hurtigere man kan skære dele af søgetræet fra, jo mere effektivt er programmet.

Tilsigtet brug

Alt efter hvordan man tilsigter brugen af sit program, kan forskellige rækkefølger af literalerne i reglerne være at foretrække. Er der kun få muligheder for en af literalerne, er det bedst at behandle denne først, da den eventuelt kan udelukke mange af mulighederne for en anden literal. Jo mere man kan bestemme ud fra tidligere trufne valg jo bedre.

Den foretrukne rækkefølge vil være forskellig, alt efter hvilke variable der er kendte og ikke kendte. Bestemmelsen af værdierne for de resterende variable, kan nemlig afhænge af værdierne for de forrige variable.

5.4 Søgestrategier

Indtil nu har vi set, at det har stor betydning, hvordan man opbygger sit program både med hensyn til effektivitet og med hensyn til terminering (søgetid) af programmet.

5.4.1 Labeling

Er propagetion og domænereduktion ikke nok til at finde en løsning, må man lede efter en mulig variabeltildeling.
Standardmetoden i CP for at tildele værdier til variable er `labeling`-prædikatet. Prædikatet forsøger at finde en tildeling, der opfylder alle de specificerede krav. Dette gøres ved først at forsøge at tildele første variabel den første værdi fra sit domæne. Derefter fortsættes med en tildeling af anden variabel og så fremdeles. Metoden udfører en pre-order dybde-først-søgning med backtracking (som beskrevet tidligere). Dette kan tage lang tid, især hvis løsningen ikke ligger først i søgetræet.

`Labeling` foretager en ukomplet søgning, idet den kun søger efter en mulig løsning. Når den har fundet en mulig løsning, vil denne blive returneret. Søgningen når altså ikke hele søgetræet igennem, medmindre løsningen ligger i det allersidste blad, eller hvis der ikke findes nogen løsning overhovedet. Dog er det muligt at lade proceduren fortsætte, hvorfra den slap, således at den kan søge efter endnu en løsning. Derved vil det være muligt, alligevel at søge hele søgetræet igennem, hvorved man vil finde alle mulige løsninger.

Det er som regel i `labeling` at det meste af tiden bliver brugt. Derfor er en effektiv søgeprocedure. lig med et effektivt program

5.4.2 Egen søgestrategi

Udover at ordne rækkefølgen af sine regler er det muligt selv at definere en søgeprocedure og dermed selv definere en strategi for, hvordan søgetræet skal opbygges.

I forbindelse med en tildeling er der to valg, der skal foretages. Man skal beslutte hvilke variable, der skal behandles først, og man skal beslutte hvilke domæneværdier, der skal testes først for variablene. Til hjælp med dette findes der flere indbyggede prædikater i ECL^{i}PS^e .

Vi vil ikke gå i detaljer med dem her, men blot nævne nogle få.

`deleteffc`(V, L, R)	Vælger den variabel V fra liste L, der har det mindste domæne og er mest begrænset.
`maxdomain`(V, Max)	Returnerer den største værdi Max i domænet for V.

`deleteffc` vælger den variabel, der har det mindste domæne, og hvis der er flere variable med samme domænestørrelse, vælges den variabel, der indgår i flest constraints. Er der stadig flere at vælge imellem, vælges den, der står først i liste L. Den liste, der opstår ved at fjerne variablen fra L, gemmes i R. Denne strategi er baseret på, at det er bedst at behandle de mest begrænsede variable først. Har vi for eksempel følgende lille program

```
dele2(V,X,Y,Z,Res):-
          [X,Y,Z]::1..5,
          X#>1,
          X#<Y,
          deleteffc(V,[X,Y,Z],Res).
```

kan man se, at `Y` er den variabel med det mest begrænsede domæne, da den skal være større end `X` og resultatet bliver da også at Y, returneres i V.

```
V = V{[3..5]}
X = X{[2..4]}
Y = V{[3..5]}
Z = Z{[1..5]}
```

`maxdomain` bruges, når man skal finde den største værdi i et domæne. Følgende eksempel vil returnere `V=4`, da 4 er den største lovlige værdi i domænet.

```
ma(V) :- X::1..5, X #< 5, maxdomain(X,V).
```

På trods af at det er muligt at forbedre programmet ved at bygge det rigtigt op og lette søgningen ved hjælp af indbyggede prædikater, vil systemet stadig foretage en dybde-først-søgning. Denne søgning vil være komplet eller ukomplet, alt efter hvad man har specificeret.

5.4.3 Søgestrategier i ECL^{i}PS^e

Det er også muligt at knytte en omkostning til en løsning og få søgeproceduren til at evaluere løsningens pris. Prisen kan så sammenholdes med en ny gennemsøgning af træet, hvor der er tilføjet et yderligere krav, om at den fundne løsning skal være bedre end den før fundne pris. Dette svarer til at man har en objektfunktion, og at man hele tiden forsøger at gøre løsningen bedre.

I ECL^{i}PS^e er der indbyggede rutiner, som benytter sig af *branch-and-bound*-teknikker til at søge løsningsrummet (søgetræet) igennem. Disse teknikker holder øje med værdien af den hidtil fundne løsning, og for hver ny variabeltildeling vurderes den nuværende omkostning. Er omkostningen ikke bedre, forkastes løsningen, og den relevante del af søgetræet skæres fra.

Disse teknikker og søgninger benytter stadig en dybde-først-søgning, hvilket nogle gange kan være dyrt, både i tid og plads. Udover de indbyggede rutiner og regelordning kan man benytte sig af såkaldte deterministiske valg.

5.4.4 Deterministiske valg

Et deterministisk valg går ud på, at når først det er foretaget, kan man udelukke andre valg og dermed skære store dele af søgetræet fra. Et eksempel herpå er brugen af `if-then-else` udtryk. Når først

`if`-udtrykket er evalueret lad os sige til sand, er det aldrig nødvendigt at gå ned i `else`-delen og hele den del af søgetræet kan skæres fra. Dog skal man være forsigtig med brugen af `if-then-else`, hvis ikke alle de indgående parametre i `if`-udtrykket er sat fast. Eventuelle frie variable vil nemlig forsøges sat fast under evalueringen af `if`-delen. Fastsættelsen sker ved at forsøge at finde en værdi for variablen, der evaluerer til sand. Denne holdes så fast fremover, hvilket kan give problemer, da man derved får taget en beslutning for nogle variable, hvor det ikke var hensigten.

En anden mulighed for at beskære søgetræerne, er hvis alle løsninger er lige gode, og der blot ønskes en mulig løsning. Til dette kan man benytte prædikatet `once`, der standser når den første er fundet. Dette sparer tid, ved ikke at undersøge flere muligheder.

5.4.5 Heuristikker

Det er også muligt at udvikle kendte heuristikker i CP. I [10] er der et eksempel i ECL^{i}PS^e på, hvordan man kan bygge en *taboo-search* til et knapsack problem. Det er dog ikke et af de første skridt man tager, både fordi der er mange parametre, der skal sættes men også fordi programmeringen er meget omfattende og problemspecifik.

5.5 Et eksempel

Vi vil i dette afsnit give et eksempel på CP brugt til løsningen af et konkret problem. Vi betragter det N-dronningeproblemet og ser problemet formuleret både i CP og i en lineær matematisk model.

5.5.1 N-dronningeproblemet

N-dronningeproblemet har været kendt i over 200 år. I starten betragtede man kun 8-dronningeproblemet, men allerede i 1850 blev problemet generaliseret til at gælde N dronninger i stedet.

Som tidligere beskrevet går N-dronningeproblemet ud på at placere N dronninger på et $N \times N$ stort skakbræt på en sådan måde, at ingen af dronningerne kan slå hinanden. Der må altså ikke stå to brikker i samme række, samme søjle, eller på samme skrå linje parallel med diagonalen.

Problemet er sværere, end det umiddelbart lyder. Antallet af forskellige muligheder for at placere de N dronninger stiger markant, for hver gang N gøres større.

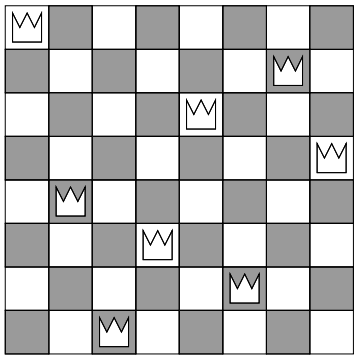
På et $N \times N$ skakbræt, er der $\binom{N^2}{N}$ muligheder for at placere N dronninger. Indføres der en begrænsning om, at der kun må være en dronning pr. række, vil der være N^N muligheder. Tilføjer man yderligere begrænsningen, at der kun må være en dronning i hver søjle, er der $N!$ muligheder.

I tabel 5.1 ses antallet af muligheder, samt antallet af løsninger, for $N = 1, 2, \ldots 10$. Antallet af løsninger er fundet ved hjælp af ECL^{i}PS^e . Det skal bemærkes at antallet, af muligheder og antallet af løsninger ikke direkte kan sammenlignes. Ud for antallet af løsninger er der taget højde for roteringer af skakbrættet, det vil sige nogle løsninger er skåret fra. Dette er ikke tilfældet ud for antallet af muligheder. Tallene giver dog stadig en rimelig fornemmelse af, hvor få løsninger der er, i forhold til hvor mange muligheder der findes.

N	$\binom{N^2}{N}$	N^N	$N!$	Løsninger
1	1	1	1	1
2	6	4	2	0
3	84	27	6	0
4	1820	256	24	2
5	53130	3125	120	10
6	1947792	46656	720	4
7	85900584	823543	5040	40
8	4426165368	16777216	40320	92
9	260887834350	387420489	362880	352
10	17310309456440	10000000000	3628800	724

Tabel 5.1: Løsninger

Figur 5.5 viser en løsning til N-dronningeproblemet for $N = 8$.



Figur 5.5: Løsning af N-dronningeproblemet, (N=8)

5.5.2 Løsning af problemet ved hjælp af. CP

Selve problemet kan beskrives på forskellige måder, alt efter hvordan man vælger at repræsentere brættet.

I CP er der mulighed for at repræsentere brættet som en liste af søjler. For tilfældet $N = 8$ har listen altså 8 elementer. Hvert element skal tildeles en værdi mellem 1 og N, og værdien svarer til den række, som dronningen er placeret i. En værdi på i for det j'te element svarer altså til, at der står en dronning på den i'te række i den j'te søjle.
Ved denne repræsentation får man automatisk inkluderet en begrænsning, der sikrer, at der kun er en dronning i hver søjle, hvilket reducerer antallet af muligheder til N^N.

Nedenfor er et CP-program der løser N-dronningeproblemet.

```
1  :- lib(fd).
2  queens_lists(N, Board) :-
3
4   length(Board, N),
5   Board :: 1..N,
6
7   (fromto(Board,[Q1|Cols], Cols, []) do
8     (foreach(Q2,Cols),param(Q1),count(Dist,1,_) do
9      Q2 #\= Q1,
10     Q2 - Q1 #\= Dist,
```

```
11        Q1 - Q2 #\= Dist
12      )
13    ),
14
15    labeling(Board).
```

Den første linje `:- lib(fd)` inkluderer et bibliotek, der indeholder procedurer til at behandle endelige domæner ("finite domain library"), hvilket netop er, hvad der arbejdes med i dette eksempel.

Prædikatet `queens_lists(N,Board)` (linje 2-13) er selve programmet til N-dronningeproblemet. Her er `N` antallet af søjler, og `Board` er listen, der i sidste ende vil indeholde løsningen til problemet.

`length(Board, N)` bruges til at oprette en liste, `Board` med længden `N`. Dette er i øvrigt et tydeligt eksempel på et prædikat, der virker "begge veje". Hvis `Board` er defineret og `N` ikke defineret, vil funktionen i stedet sætte `N` lig med antallet af elementer i listen. Hvis både `Board` og `N` er defineret, vil funktionen kunne bruges til at fortælle, om der er `N` elementer i `Board` eller ej.

`::` er en funktion fra biblioteket `fd`, som blev inkluderet i linie 1. Funktionen definerer domænerne for variablene. `Board :: 1..N` begrænser altså domænerne for elementerne i `Board` til at være heltal fra mængden $\{1, \ldots, N\}$.

`fromto(First,In,Out,Last)` er en indbygget funktion, der virker lige som en for-løkke. Den starter med at sætte `In=First` og udfører, det den skal, indtil `Out=Last`.
Linjen `(fromto(Board,[Q1|Cols], Cols, []) do`, er starten på en løkke, der rækker seks linjer frem. Inde i løkken angives de begrænsninger, der skal overholdes, for at en tildeling af værdier til variablene er en løsning.
I dette tilfælde tager vi det første element i `Board` og fortsætter, indtil vi har været hele listen igennem, det vil sige indtil elementet, efter det vi ser på, er den tomme liste. Undervejs udføres indholdet af `fromto`-løkken, som er de efterfølgende 4 linjer.

`foreach(Q2,Cols)` lader variablen `Q2` løbe over hele listen `Cols`. I dette tilfælde kigger vi på hvert element `Q2` i den del af listen, der ikke er blevet behandlet endnu
De argumenter, der står inde i `param`, er blot de parametre, det er nødvendigt at kende inde i løkken. Parametrene udenfor en løkke følger ikke automatisk med ind i den.

Dernæst benyttes `count(I,Min,Max)`, som lader `I` løbe fra `Min` til `Max`. I dette tilfælde betragter vi afstanden `Dist` mellem to søjler. `Dist` starter med at være en og bliver en større for hvert skridt, vi tager.

Nu kan vi angive præcist, hvad det er, vi vil passe på. Den første begrænsning udtrykker, at ingen af tallene må være ens. Dette er et udtryk for, at der ikke står mere end én dronning i hver række. De næste to begrænsninger sikrer, at dronningerne ikke står på samme skrå linje parallel med diagonalen. Da `Dist` altid er positiv, ser vi på forskellen mellem søjleværdierne begge veje.

Til sidst kaldes den indbyggede procedure `labeling`, der forsøger at tildele værdier til variablene, samtidig med at begrænsningerne overholdes. Når en mulig løsning er fundet, returneres denne.

5.5.3 Matematisk model

Hvis vi betragter N-dronningeproblemet i forhold til matematisk modellering, er der flere forskellige måder at beskrive det på. Vi har her valgt at beskrive skakbrættet som en $N \times N$ matrix. Hver plads $x_{i,j}$ i matricen svarer til et felt på skakbrættet. Hvis der står en dronning i række i, kolonne j, vil $x_{i,j} = 1$ og ellers lig 0.

Dette giver en model, der ser ud som følger.

$$\max \sum_{i}^{N} \sum_{j}^{N} x_{i,j} \qquad \text{uht.} \tag{5.4}$$

$$\sum_{i=1}^{N} x_{i,j} \leq 1 \qquad j \in [0; N] \tag{5.5}$$

$$\sum_{j=1}^{N} x_{i,j} \leq 1 \qquad i \in [0; N] \tag{5.6}$$

$$\sum_{i=1}^{N-k} x_{i+k,i} \leq 1 \qquad k \in [0; N-2] \tag{5.7}$$

$$\sum_{i=1}^{N-k} x_{i,i+k} \leq 1 \qquad k \in [1; N-2] \tag{5.8}$$

$$\sum_{i=1}^{k-1} x_{k-i,i} \leq 1 \qquad k \in [3; N+1] \tag{5.9}$$

$$\sum_{i=k}^{N} x_{i,N+k-i} \leq 1 \qquad k \in [2; N-1] \tag{5.10}$$

Da man skal maksimere antallet af dronninger og samtidig overholde, at der højst må være en dronning i hver søjle og hver række ((5.5) og (5.6)), ender man med at bruge præcis N dronninger. De sidste fire begrænsninger sikrer, at der højst står en dronning på hver skrå linje. Man kigger på de skrå nedadrettede linjer fra venstre mod højre over og under diagonalen, og derefter på de skrå opadrettede linjer over og under diagonalen.

5.5.4 Opsamling

Som man kan se er det muligt at skrive et ret lille program i CP og få det til at løse N-dronningeproblemet. Der er ikke megen indeksering, og selve repræsentationen er noget mindre og mere overskuelig end i den matematiske model. Ved små kørsler af N-dronningeeksempler i de to sprog fandt vi at GAMS var hurtigere til at finde en løsning især for store N. Benyttede man sig af heuristiske søgestrategier i ECL^{i}PS^e var de omtrent lige hurtige. På hjemmesiden for ECL^{i}PS^e [10] kan man finde et eksempel på et N-dronningeproblem, modelleret med flere forskellige søgestrategier. På hjemmesiden for GAMS [9], kan man også finde en modellering af N-dronningeproblemet, og det er disse to modeller, vi har brugt til vores kørsler. De to programmer kan ses i bilag B.

5.6 Styrker og svagheder ved CP

Vi har i det foregående givet et overblik over CP og mulighederne heri. Nu vil vi komme ind på nogle af vore betragtninger omkring CP i forhold til andre programmeringssprog.

Som vi har set, er CP mere fleksibelt end mange andre programmeringssprog. I eksemplet med Newtons 2. lov viste vi, hvordan en defineret relation kan benyttes på mange forskellige måder. Det er lige meget hvilken variabel, vi ønsker at finde. I almindelige programmeringssprog ville dette kræve forskellige funktioner, alt efter hvilke parametre der var kendte.

En regel kan også bruges til at verificere tildelte variable. Det vil sige, den kan give et sand/falsk-svar eller et talsvar $X = 2$ tilbage alt efter den ønskede brug. Dette er ikke muligt i mange andre

sprog, såsom `C`, `Java` og `Pascal`. Ligeledes er det heller ikke altid nødvendigt, på forhånd at typebestemme inddata- og uddata-variablene. I mange sprog skal man direkte definere hvilken type inddata-variable man skal benytte og hvilken type uddata man ønsker.

Denne fleksibilitet gør, at man skal være mere opmærksom, når man definerer regler. På grund af en regels alsidighed er der flere tilfælde, hvor søgningen efter en løsning kan gå galt. Man skal være meget opmærksom på rækkefølgen og indholdet af reglerne, så man kan sikre sig at programmet terminerer.
Problemet med manglende terminering støder man også på i andre programmeringssprog. For eksempel i forbindelse med forkert opbyggede `for`-løkker og gensidige funktionskald. Dog er der i de andre sprog kun én slags brug og dermed kun en fremgangsmåde, så det kan være lettere at sikre, at programmet terminerer.

Det er lettere at definere regler, relationer og problemer i CP end i mange andre sprog, men det kræver samtidig et vist overblik, hvis problemstillingen ikke er simpel. Laver man reglerne for generelle, begrænser de ikke problemet, og laver man dem for snævre, udnytter man ikke den underliggende solvers kapacitet og kan risikere at skære gode løsninger fra.

CP-programmer er tit mindre gennemsigtige med hensyn til hvad der egentlig foregår under kørslen. Dette er tilfældet, fordi man “bare” definerer problemet. Derefter overlades det hele til en “sort boks” (solveren), der løser problemet. I andre sprog definerer man skridt for skridt, hvad der skal gøres. Umiddelbart kan man sige, at brugen af de andre sprog kræver større matematisk indsigt i problemet, og for simple problemstillinger er dette nok også sandt. Dog skal man stadig tænke over repræsentationen af problemet i CP, og i begge tilfælde skal man finde en passende model.

Lineære ligningsløsere kræver lineære formler, derfor må man tit tilpasse problemerne, så de kan lineariseres. CP kræver ikke en linearisering for at opbygge et søgetræ, og derfor er det til tider muligt at lave en bedre model af virkeligheden. Man skal dog være opmærksom på, at dette kan koste i forhold til søgetid og kvalitet af løsningen.

Ved lidt større problemer kan løsningsrummene og søgetræerne hurtigt blive for store til, at det er hensigtsmæssigt blot at overlade dem til solveren. Her er det tit en stor fordel at hjælpe solveren på vej, for eksempel ved at definere ekstra regler eller ved at sørge for, at søgetræet bliver opbygget hensigtsmæssigt. Dette har både fordele og ulemper. Ved at hjælpe solveren lidt kan man få mere effektive programmer og udnytte teknikken til at finde hurtige løsninger.
Når man begynder at opstille søgeregler og ekstra begrænsninger, gør man samtidig selve programmet mere problemafhængigt og mindre generelt. Det er ikke sikkert, at en god søgemetode til en instans af problemet nødvendigvis er en god metode til alle instanser af problemet.

Det er værd at nævne, at det stadig er forholdsvis begrænset, hvad der findes af litteratur i forbindelse med CP. Der findes en del vedrørende teorien bag logikprogrammering, kunstig intelligens og CSP. Men det har ikke været let at finde materiale til brug for udvikling af CP-modeller i praksis. Heri ligger en helt praktisk ulempe i forhold til mere etablerede paradigmer.

Kapitel 6

Constraint programming model

I dette kapitel vil vi beskrive, hvorledes vi har modelleret vagtplanlægningsproblemet i CP. Modellen opfylder de krav, der blev sammenfattet i afsnit 3.2.

Vi indleder kapitlet med en kort oversigt over hvilke inddata og uddata, der benyttes. Herefter gennemgås modelleringen af begrænsningerne, samtidig med at vi fremhæver visse interessante elementer i modelleringen. I forbindelse med beskrivelsen af de bløde begrænsinger, kommer vi desuden ind på hvorledes objektfunktionen opbygges. I afsnit 6.4 gennemgår vi vores søgeprocedure. Det endelige CP-program kan ses i bilag C.1.

6.1 Overblik

Som vi har beskrevet før, skal modellen være i stand til at lægge en vagtplan ud fra et givet behov og en given personalemængde.

Hovedpunkterne i programmet er følgende. Først sørger modellen for at fastlægge de begrænsninger, der blev fastlagt i kapitel 3. De hårde begrænsninger defineres således, at de altid skal overholdes, og de bløde begrænsninger tilknyttes en pris for eventuelle overtrædelser. Den samlede omkostning for at overtræde en blød begrænsning overføres til objektfunktionen.
Når begrænsningerne er definerede, sker den egentlige tildeling af værdier til modellens variable. Til dette benytter vi et *branch-and-bound*-prædikat, samt en decideret søgemetode, vi har konstrueret til dette problem.

6.1.1 Notation

Vi vil her beskrive lidt af den notation, der er blevet brugt i vores model. Vi har valgt kun at beskrive de parametre og variable, der er relevante i forbindelse med forståelsen af den overordnede modellering af problemet.

NNurses
: Det samlede antal sygeplejersker planen lægges for.

NDays
: Det samlede antal dage planen lægges for.

NLabels=4
: Summen af antallet af arbejdstyper (dag, aften, nat) plus en.

NShifts=5
Det samlede antal vagttyper (dag, aften, nat, fri, sove).

DayC, LabelC, NurseC
Variable der benyttes som tællere inde i diverse løkker.

Schedule[NNurses,NDays]
Matricen indeholdende den fundne løsning. Hver række repræsenterer en sygeplejerske, og hver søjle repræsenterer en dag. Indholdet i cellerne indikerer, hvilken vagttype den pågældende sygeplejerske har den aktuelle dag.

Dem[NLabels,NDays]
En matrix indeholdende det definerede behov. Summen af hver søjle skal svare til det samlede antal sygeplejersker. Hver celle indeholder antallet af sygeplejersker, der er behov for til den pågældende vagt. Her er fri- og sovevagter slået sammen til en række.

Employees[4,NNurses]
En matrix indholdende data vedrørende hver sygeplejerske. Hver søjle svarer til en sygeplejerske. I de fire rækker aflæses hhv. navn, nattevagtsmulighed, timetal og fridagsønsker.

'x' of shift
Struktur der benyttes i forbindelse med vagttyper:

Dagvagt	d of shift
Aftenvagt	e of shift
Nattevagt	n of shift
Fridag	f of shift
Sovedag	s of shift

I programmet svarer hver vagttype til et heltal mellem 1 og `NShifts`. I stedet for at betegne en dagvagt med '1', er det nu muligt at kalde denne for `d of shift`.

Det skal desuden bemærkes at vi igennem hele programmet arbejder med heltalsdomæner. Dette betyder, at når vi for eksempel begrænser en variabel til at være indeholdt i intervallet `[1,...,NShifts]`, så kan den kun antage heltalsværdierne indenfor dette interval. Ved hjælp af den ovenfor beskrevne struktur (`'x' of shift`) svarer det til, at variablen kun kan have værdien `d of shift`, `e of shift`, `n of shift`, `f of shift` eller `s of shift`.

6.1.2 Inddata

Modellens inddata består af matricen `Dem[NShifts,NDays]`, der definerer behovet, samt matricen `Employees[4,NNurses]`, der definerer arbejdsstyrken.

For at få en løsning er det er en forudsætning, at der hver dag er personale nok i forhold til arbejdsbehovet og kravet om hviletidsperioder.

6.1.3 Uddata

Resultatet af den fundne løsning kan aflæses i matricen `Schedule[NNurses, NDays]`. Når den endelige løsning er fundet, bliver vagtplanen skrevet ud på skærmen. I figur 6.1 har vi vist et udklip af en sådan vagtplan. Resten af uddata kan ses i bilag C.1.6.

```
       m  t  o  t  f  l  s  m  t  o  t   ...   m  t  o  t  f  l  s |  D  A  N | AV WS FØ EV  sum |
-------------------------------------------   -------------------|---------|-----------------|
Nr. 1:  N  S  F  F  D  A  A  F  D  F  F   ...   A  F  F  D  D  F  F |  6  5  2 |  0  0  0  0   0  |
Nr. 2:  F  N  S  F  F  A  A  F  D  D  D   ...   A  F  D  D  F  F  F |  5  6  2 |  0  0  0  0   0  |
Nr. 3:  D  F  N  S  F  D  D  F  A  A  F   ...   F  F  F  A  A  F  F |  6  5  2 |  0  0  0  0   0  |
Nr. 4:  F  D  F  N  S  D  D  F  A  A  A   ...   F  F  F  A  F  F  F |  5  4  2 |  0  0  0  0   0  |
Nr. 5:  D  F  A  F  F  F  F  D  F  D  F   ...   D  A  F  F  F  A  A |  5  6  0 |  0  0  0  0   0  |
-------------------------------------------   -------------------|---------|-----------------|
Nr. 6:  F  F  F  F  F  F  F  D  F  F  F   ...   N  S  F  F  F  A  A |  3  4  1 |  0  0  0  0   0  |
Nr. 7:  F  A  F  A  A  F  F  N  S  F  F   ...   F  N  S  F  F  D  D |  5  4  2 |  0  0  0  0   0  |
Nr. 8:  F  A  F  F  F  F  F  A  N  S  F   ...   F  F  N  S  F  D  D |  4  3  2 |  0  0  0  0   0  |
Nr. 9:  F  F  D  D  N  S  F  A  F  F  F   ...   S  D  F  F  A  F  F |  3  4  3 |  0  0  0  0   0  |
Nr.10:  A  F  D  D  A  F  F  F  F  N  S   ...   D  F  A  F  F  F  F |  3  4  3 |  0  0  0  0   0  |
-------------------------------------------   -------------------|---------|-----------------|
Nr.11:  A  F  F  A  F  N  N  S  F  F  D   ...   F  D  A  F  D  F  F |  4  4  3 |  0  0  0  0   0  |
Nr.12:  F  F  F  F  D  F  F  F  F  F  A   ...   F  A  F  F  N  N  N |  4  4  3 |  0  0  0  0   0  |
Nr.13:  F  D  A  F  F  F  F  F  F  F  N   ...   F  F  D  N  S  F  F |  3  3  3 |  0  0  0  0   0  |
-------------------------------------------   -------------------|---------|-----------------|
AV: 'Antal vagter'
WS: 'Weekendsammenlægning'
FØ: 'Fridagsønsker'
EV: 'Efterfølgende vagter'

Samlede pris: 0
Strategi: step
Benyttet tid: 68.3
```

Figur 6.1: Eksempel på uddata fra programmet (af pladshensyn er midten skåret fra)

6.2 Hårde begrænsninger

I det følgende gennemgår vi de hårde begrænsninger.

Én vagttype pr. dag

Som beskrevet ovenfor, opbygger modellen en matrix `Schedule[NNurses, NDays]` indeholdende den færdige vagtplan. Hver celle $[nn, nd]$ svarer til en variabel, der bestemmer vagttypen for sygeplejerske nn på dag nd.

Når skemaet oprettes, angives et domæne for hver af variablene. Den første dag i perioden er et specialtilfælde, da denne ikke kan være en sovedag. Til dette tilfælde benytter vi `NLabels` i stedet for `NShifts`, da `NLabels` netop svarer til antallet af vagter minus sovedagsmuligheden. Domænespecifikationerne ser ud som følger:

```
Schedule[NurseC,1] :: 1..NLabels
Schedule[NurseC,DayC] :: 1..NShifts
```

Her benyttes prædikatet `::`, som sørger for, at variablen `Schedule[NurseC, DayC]` på venstre side bliver bundet af domænet defineret på højre side `[1,...,NShifts]`.

I den færdige vagtplan har hver variabel en bestemt værdi. Denne modellering medfører direkte at hver sygeplejerske har én vagttype pr. dag.

Behovet opfyldes præcist

Det næste skridt, er at sørge for at bemandingsbehovet bliver opfyldt. Behovet er defineret i behovsmatricen `Dem[NLabels,NDays]`. I programmet ser vi på en arbejdstype og en dag ad gangen, og vi benytter os af følgende kode:

```
Required is Demand[LabelC,DayC],
Shifts is Schedule[1..NNurses,DayC],
occurrences(LabelC,Shifts,Required)
```

I første linie sætter vi `Required` lig med behovet for arbejdstype `LabelC` dag `DayC`. Dernæst tager vi søjlen i vagtplansmatricen, for dag `DayC`, og lægger denne i `Shifts`.

Til sidst benytter vi prædikatet `occurrences`, som kræver, at antallet af variable med værdien `LabelC` i listen `Shifts` er lig med `Required`.

Koden implementeres ved hjælp af forskellige løkker, så vi kommer alle behovene på alle dagene igennem.

Fri hver anden weekend

For at give hver medarbejder fri hver anden weekend går vi ind i en `for`-løkke. Her ser vi på hver medarbejder og på to fortløbende uger ad gangen. Vi piller lørdage og søndage ud for to uger og kræver, at mindst en af weekenderne har frivagter begge dage. Kravet modelleres ved følgende kode:

```
(Sat1 #= f of shift #/\ Sun1 #= f of shift) #\/
(Sat2 #= f of shift #/\ Sun2 #= f of shift)
```

`Sat1`, `Sun1`, `Sat2` og `Sun2` er variable, der udtrykker vagttyperne for lørdagene og søndagene i to fortløbende weekender. Prædikaterne `#=` , `#\/` og `#/\` svarer til de tilsvarende symboler fra matematisk logik ($=$, $\vee$ og $\wedge$).

Fridøgnsperioder

Modellen skal naturligvis overholde de definerede fridøgnsperioder for alle medarbejderne. Vi kigger på en medarbejder og en uge ad gangen. For hver uge ser vi nu på tre fortløbende dage for at undersøge, om nogen af kombinationerne for fridøgnsperiode er overholdt. Er en fridøgnsperiode overholdt, evaluerer `FD1`, `FD2`, `FD3` eller `FD4` til 1, ellers til de 0. Summen af disse gemmes i en variabel.

```
FreeDays1 #= FD1+FD2+FD3+2*FD4
```

`FD4` indikerer om der er nogen lange fridøgnsperioder (mindst 55 timer), derfor multipliceres denne med 2. `FreeDays1`-elementet lægges ind i en liste `FreeDays`. Resten af tilfældene for tre sammenhængende dage i den pågældende uge lægges tilsvarende ind i `FreeDays`. Summen af elementerne i listen skal mindst være 2, da man så har enten to korte eller en lang fridøgngsperiode. Dette udtrykkes som:

```
sumlist(FreeDays,SumFD),
SumFD #>= 2
```

`sumlist(FreeDays,SumFD)` summerer elementerne i `FreeDays` og lægger resultatet i `sumFD`.

Alt dette sammenkædes ved hjælp af nogle `for`-løkker og `if-then-else`-sætninger, som for så vidt ikke er interessante i denne sammenhæng, og derfor ikke vises her.

En fridag pr. 7 dage

Alle medarbejdere skal have mindst en fridag pr. løbende 7 dage. Dette svarer til, at hver medarbejder har mellem 1 og 7 fridage for hver løbende 7 dage:

```
Days is Schedule[NurseC,DayC..DayC+6],
occurrences(f of shift,Days,DaysOff),
DaysOff :: 1..7
```

Da `DayC` er en tæller, som løber fra dag 1 og frem til den syvende-sidste dag i perioden, vil `Days` løbende indholde alle syvdages-perioder af vagtplanen.
I anden linje benytter vi igen prædikatet `occurrences`, som sørger for, at antallet af fridage i den aktuelle periode bliver knyttet til variablen `DaysOff`. I sidste linie sørger vi for at binde denne variabel til domænet $[1, \ldots, 7]$, hvorved det ønskede er opnået.

Nattevagter

Antallet af nattevagter, hver medarbejder skal have, bliver lagt fast, inden modellen prøver at fordele de resterende vagttyper.

```
sumlist(OccOfNightShifts,NNightShifts),
once labeling(OccOfNightShifts),
```

`OccOfNightShifts` er en liste med en variabel for hver sygeplejerske. Variablene indikerer, hvor mange nattevagter sygeplejerskerne skal have. Variablenes domæner er de øvre og nedre grænser for antallet af nattevagter for hver sygeplejerske. Disse beregnes på samme måde som i den matematiske model. Første linje udtrykker, at summen af variablene i listen skal svare til behovet for nattevagter `NNightShifts`. Anden linje kalder `labeling`-prædikatet, der forsøger at finde en tildeling. `once` angiver, at vi er tilfredse med den første løsning der bliver fundet. Her er en klar fordel i forhold til den matematiske model beskrevet tidligere. Her bestemmer vi allerede nu antallet af mattevagter til hver sygeplejerske og kender dermed det præcise tal til senere beregninger.

Sovedagsrestriktioner

Efter en nattevagt må det kun være muligt at få endnu en nattevagt eller at få en sovedag. Desuden må sovedage kun findes efter nattevagter.

Vi behandler igen én medarbejder ad gangen. For denne medarbejder ser vi på alle tilfælde af to fortløbende dage og kalder disse dage for henholdsvis `SomeDay` og `NextDay`:

```
SomeDay #= n of shift #/\
          NextDay::[n of shift, s of shift] #\/
SomeDay #\= n of shift #/\ NextDay #\= s of shift,
```

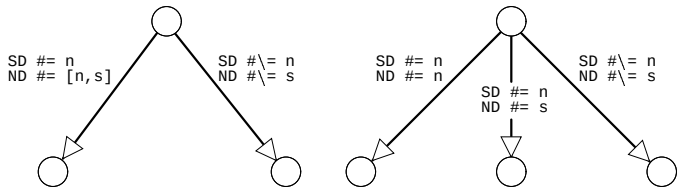
Her introducerer vi endnu et prædikat #\= , som svarer til $\neq$.
De første linjer sørger for, at hvis `SomeDay` er en nattevagt, så indskrænkes domænet for `NextDay` til at være enten en nattevagt eller en sovedag. Den sidste linje sørger for, at hvis `SomeDay` ikke er en nattevagt, så kan `NextDay` ikke være en sovedag. Da en af disse to linier altid skal være overholdt, er det ønskede nu opnået.

Denne begrænsning kan også udtrykkes anderledes.

```
SomeDay #= n of shift #/\ NextDay #= n of shift  #\/
SomeDay #= n of shift #/\ NextDay #= s of shift
SomeDay #\= n of shift #/\ NextDay #\= s of shift
```

Ulempen ved at bruge disse tre linjer i stedet for de to, vi har benyttet, er, at søgetræet vil blive større.

Ved at bruge sidstnævnte metode vil en beslutning om at sætte en vagt til en nattevagt medføre, at der med det samme skal tages et valg for, om den efterfølgende vagt skal være en nattevagt eller en sovevagt.



Figur 6.2: Søgetræets størrelse

I førstnævnte metode vil en beslutning om at sætte en vagt til en nattevagt ikke medføre, at der skal foretages et valg med det samme. I stedet indskrænkes domænet for den variabel, der betegner den efterfølgende vagttype, som sagt til enten at være en nattevagt eller en sovevagt.

11-timersreglen

Som vi har beskrevet i afsnit 4.2, er 11-timersreglen stort set automatisk opfyldt, da man kun kan have en vagt pr. dag. Det eneste, der skal tilføjes, er en begrænsning, der sikrer, at en medarbejder ikke har dagvagt dagen efter en aftenvagt.

Fremgangsmåden til dette minder meget om den metode, vi brugte til at udtrykke sovedagsrestriktionerne. Vi arbejder igen med `SomeDay` og `NextDay`, som tilføjes følgende begrænsning:

```
SomeDay #= e of shift #/\ NextDay #\= d of shift #\/
SomeDay #\= e of shift
```

Som det ses, er det nu ikke muligt at have en dagvagt efter en aftenvagt.

6.3 Bløde begrænsninger

For at modellere de bløde begrænsninger benytter vi os af prædikatet:

```
B isd C ,
```

hvor `B` er en binær variabel, der udtrykker, om begrænsningen `C` er opfyldt eller ej. `B` er lig med 1, hvis `C` er opfyldt og 0 ellers. Herved er det let at teste, om de bløde begrænsninger er overholdt. Værdien af `B` multipliceres med prisen for den aktuelle bløde begrænsning, og resultatet lægges til den samlede omkostning.

I det følgende vil vi gennemgå de fire bløde begrænsninger. Alle fire begrænsninger bliver behandlet inde i én samlet `for`-løkke, som køres for en medarbejder ad gangen. Begrænsningerne defineres altså for hver medarbejder.

Vagter deles ligeligt

Modellen skal forsøge at fordele vagterne i forhold til de ansattes timetal. Vi går frem som beskrevet i afsnit 3.1.5, hvor vi fandt følgende interval for antallet af vagter:

$$[\lceil X_{s,a} \rceil - 1; \lceil X_{s,a} \rceil] \ . \tag{3.6}$$

Fordelen ved CP-modellen set i forhold til den matematiske model er, at vi kender det nøjagtige antal timer, hver medarbejder har nattevagter. Tallet h'_s angiver derfor nøjagtig det antal timer, hver medarbejder har til overs til dag- og aftenvagter.

For hver medarbejder finder vi nu et tal for hver vagttype. Tallet angiver, hvor mange vagter medarbejderen har for meget eller for lidt i forhold til de bløde grænser. Dette tal overføres til de samlede omkostninger.

Fridagsønsker

For hver medarbejder kan man angive eventuelle fridagsønsker. Som vi har været inde på tidligere, gøres dette ved hjælp af en liste, der indeholder en parameter for hver dag i vagtplanen. Hvis værdien af en af parametrene er større end 0, svarer det til, at medarbejderen ønsker fri den pågældende dag. Jo større værdien er, jo større er ønsket om at få fri.

For hver medarbejder går vi nu frem en dag ad gangen og ser på fridagsønskerne. Fridagsønskeparametrene sammenholdes med vagttypen den pågældende dag. I koden herunder er `SomeDay` vagttypen og `SomeHoliday` er fridagsønskeparameteren.

```
ViolHoliday1_tmp isd
          SomeDay #\= f of shift #/\
          SomeHoliday #\= 0,

ViolHoliday1 #= ViolHoliday_tmp*SomeHoliday
```

Resultatet er, at den binære variabel `ViolHoliday_tmp` er lig 1, hvis både `SomeDay` ikke er en frivagt, og `SomeHoliday` er større end 0. Det vil sige, hvis der arbejdes på en dag, der ønskes fri. I sidste linie multiplicerer vi den binære variabel med fridagsønskeparameteren, og resultatet lægges

i `ViolHoliday`. Det bemærkes her, at `Violholiday` netop vil være lig 0, hvis man enten har fået en fridag, eller hvis man ikke specifikt har ønsket en fridag.

Værdien af `ViolHoliday`, som nu afspejler, om et ønske om fridag er blevet overholdt eller ej, overføres til de samlede omkostninger.

Weekendvagter

For hver medarbejder skal modellen forsøge at tildele samme type vagt for lørdag og søndag i samme weekend. I denne sammenhæng svarer en sovedag til en fridag.
Vi ser nu på vagterne i en weekend ad gangen. Variablen `Sat` svarer til lørdagsvagten og `Sun` svarer til søndagsvagten. Følgende udtryk sørger for, at variablen `ViolWeekend` er lig 1, hvis begrænsningen er overtrådt og 0 ellers. Værdien af denne variabel overføres til den samlede omkostning.

```
ViolWeekend isd
   (Sat #= d of shift #/\ Sun #\= d of shift ) #\/
   (Sat #= e of shift #/\ Sun #\= e of shift ) #\/
   (Sat #= n of shift #/\ Sun #\= n of shift ) #\/
   ((Sat #= f of shift #\/ Sat #= s of shift )
     #/\ Sun #\= f of shift)
```

Vagtrækkefølge ([sove,nat])

Modellen skal forsøge at undgå at tildele en medarbejder nattevagter dagen efter, at denne har en sovevagt. Vi ser derfor på to på hinanden følgende dage og sætter variablen `ViolSubS` lig 1, hvis `SomeDay` er en sovedag samtidig med, at `NextDay` er en nattevagt:

```
ViolSubS isd
         SomeDay #= s of shift #/\
         NextDay #= n of shift
```

Igen overføres `ViolSubS` til de samlede omkostninger.

6.3.1 Samlede omkostninger

Modellen skal forsøge at overholde alle de bløde begrænsninger. Udtrykkene for de bløde begrænsninger evaluerer til en værdi, der fortæller, om begrænsningen er overholdt eller ej. En værdi på 0 svarer til, at begrænsningen er overholdt, og en værdi større end 0 svarer til, at begrænsningen er overtrådt. Jo større værdien er, jo værre er overtrædelsen.

Hvis det ikke er muligt at overholde alle de bløde begrænsninger, ønsker vi, at modellen fordeler overtrædelserne så meget som muligt blandt medarbejderne. Det skal eksempelvis være bedre at give to medarbejdere en ekstra dagvagt hver fremfor at give én medarbejder to ekstra dagvagter. Desuden skal det være muligt at prioritere vigtigheden af de bløde begrænsninger i forhold til hinanden.

I stedet for direkte at tilføje værdierne til den samlede omkostning sørger vi for, at multiplicere med en konstant p_i, $i \in \{1, 2, 3, 4\}$, afhængig af hvilken af begrænsningerne der er tale om.
For hver medarbejder vil der altså være fire tal, som tilsammen beskriver, hvor god dennes vagtplan

er. For hver medarbejder kvadrerer vi summen af de fire tal og lægger denne værdi til i objektfunktionen.

Ved at kvadrere vokser den samlede omkostning hurtigere, hvis overtrædelserne er samlet ved en medarbejder. Eksempelvis er $(1+0+0+0)^2+(1+0+0+0)^2 = 2$ bedre end $(1+1+0+0)^2 = 4$. Dette betyder, at der er meget at vinde ved at fordele overtrædelserne mest muligt. Dette er netop, hvad vi ønsker.

Når programmet har fundet en muligt løsning udskrives denne. Samtidig returneres en værdi for, hvor god denne løsning er. Dette tal svarer til det, der blev benyttet i objektfunktionen, dog uden at det er blevet kvadreret. Derved viser det returnerede tal mere direkte, hvor mange begrænsninger, der er blevet overholdt.

6.4 Søgning

Indtil nu har vi fået opbygget en model, der indeholder en mængde variable, der tilsammen udgør en samlet vagtplan. Det, vi nu ønsker, er at finde en egentlig tildeling til alle variablene, således at hver variabel indeholder netop én værdi.

Dog er vi er ikke *kun* interesserede i at finde en vagtplan, men også i at finde en *god* vagtplan.

Prøver man at lade ECL^{i}PS^e løse modellen ved hjælp af det indbyggede `labeling`-prædikat beskrevet i afsnit 5.4.1, går det hurtigt galt.
Problemet med at bruge dette prædikat er dels, at det ikke giver mulighed for at evaluere omkostningen af en løsning i forhold til en anden, og dels at søgningen blot starter fra en ende af og derfra udfører en dybde-først-søgning.

Run-time-systemet tildeler fra en ende af en type vagter til en medarbejder. Da begrænsningen angående dag- og aftenvagter er en blød begrænsning, tages der slet ikke hensyn til denne. Den først behandlede medarbejder vil ofte få mange af den først behandlede vagttype, og de sidste medarbejdere vil tildeles få vagter, hvis nogen overhovedet. Dette fører ikke til noget fornuftigt.

Der er flere muligheder for at hjælpe søgningen på vej. Man kan tilføje hårde begrænsninger, der beskærer søgetræet yderligere. Eksempelvis ved at sætte hårde grænser for nogle af variablene og på denne måde skære værdier væk, som man ikke regner med at få brug for.

En mulighed kunne være at sætte en hård øvre grænse for antallet af dagvagter, en medarbejder må have. Ulempen er, at man risikerer at gøre problemet for begrænset og i sidste ende udelukke muligheden for overhovedet at finde en løsning.

En anden mulighed er at fortælle hvad solveren skal forsøge først, og hvorledes den skal gå videre. For eksempel kan man vælge at opbygge søgetræet således, at der først tildeles værdier til udvalgte variable, eller også sådan at der først tildeles værdier fra bestemte dele af variablenes domæner.

Derudover kan man også vælge at dele problemet op i mindre delproblemer og løse disse enkeltvis. Ulempen ved at dele problemet op i delproblemer er, at man ikke længere kan være sikker på, at man finder en den bedst mulige løsning. Den endelige løsning er stykket sammen af flere delløsninger, der hver især er bedst mulige, men som eventuelt ved sammenstykning kunne være bedre, hvis de havde andre værdier. Fordelen ved at dele problemet op i delproblemer er, at søgetiden mindskes markant, da hver del indeholder færre variable og dermed betydelig mindre søgetræer.

Vi har valgt at bruge en kombination af de to sidstnævnte muligheder. Vi indbygger en mulighed for at dele problemet op og derved løse mindre dele af problemet ad gangen. Derudover angiver vi, hvordan meget af søgningen skal gå frem. Vi definerer altså strukturen på søgetræet, med formålet at forsøge at flytte de gode løsninger over i venstre del af søgetræet

Herunder fortæller vi, hvordan vi har gjort i forbindelse med opbygningen af programmet.

6.4.1 *Branch-and-Bound*

Den styrende mekanisme i vores søgning efter en god løsning er prædikatet
`bb_min`, som er indbygget i ECL^{i}PS^e . Prædikatet forsøger at minimere den samlede omkostning ved hjælp af *branch-and-bound*-metoden. Prædikatet ser ud som følger:

$$\texttt{bb\_min}(+\texttt{Goal}, ?\texttt{Cost}, +\texttt{Options})$$

Inddata til dette prædikat er en søgemetode `+Goal`, en variabel `?Cost`, der skal minimeres og til sidst nogle *branch-and-bound*-specifikke parametre `+Options`.
Søgemetoden vil vi gennemgå i det følgende. `Cost` er summen af bidragene fra de bløde begrænsinger, som naturligt nok skal minimeres. `Options` består af flere valgfrie dele, som kan have indflydelse på, hvordan *branch-and-bound*-metoden skal foregå. Vi har valgt at benytte følgende:

Strategy: Her vælges en af følgende:

- **continue** Når en løsning er fundet, fortsætter søgningen med en ny begrænsning på `Cost`. Den nye grænse vil være lig med omkostningen på den bedst fundne løsning minus 1.
- **step** Når en løsning er fundet, starter søgningen forfra, med en ny begrænsning på `Cost`. Her vil den nye grænse også være lig med omkostningen på den bedste fundne løsning minus 1.
- **dichotomic** Når en løsning er fundet, halveres dennes omkostning, og der søges efter en løsning der har en omkostning i den nedre halvdel, hvis en sådan ikke findes, søges der videre i den øvre halvdel.

Timeout: Maksimum for den samlede søgetid
From: Nedre grænse for den samlede omkostning
To: Øvre grænse for den samlede omkostning

Et eksempel i pseudo-kode på hvordan `bb_min` kaldes:

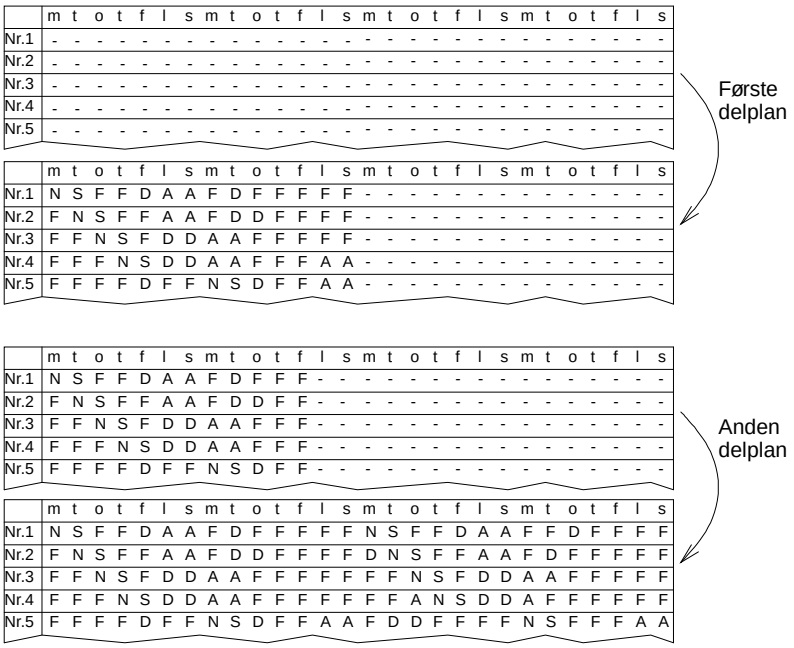
$$\texttt{bb\_min}(\texttt{labeling}, \texttt{SumCost}, \texttt{bb\_optionswith}(\texttt{step}, 1000, 0, 500))$$

Dette angiver, at vi ønsker at bruge `labeling`-prædikatet til at minimere
`SumCost`, der er summen af værdierne fra de bløde begrænsninger. Dette skal gøres ved hjælp af `step`-strategien med en søgetid på højst 1000 sekunder. Vi ønsker en løsning, der har højst har en værdi på 500 og endvidere har en nedre grænse på 0.

6.4.2 Opsplitning

Vi har indbyget den mulighed at lægge vagtplanen ved at dele perioden op i flere dele. Derved kan man eksempelvis lægge en vagtplan for 14 dage ad gangen, indtil planen for hele perioden er bestemt. Fordelen ved denne metode er, at søgningen går væsentlig hurtigere, da søgetræerne for 14 dage er lettere at arbejde med end dem for en hel måned. Ulempen er, at man mister overblikket, som kan fås ved at lægge hele planen på en gang. Dog mindskes dette tab en del, idet modellen altid arbejder med en vagtplan fra dag 1 og frem. Det vil sige, at når planen for eksempelvis dag 15 til 28 skal lægges, arbejdes der med en plan for dag 1 til 28, hvori de første 14 dage er lagt fast. Dette betyder, at en eventuel skæv fordeling af vagttyper i løbet af de første 14 dage, vil forsøges at blive rettet op i løbet af de efterfølgende 14 dage.

For at kompensere yderligere for det manglende overblik der opstår ved at dele perioden op i flere dele, har vi formuleret CP-modellen således, at man for hvert gennemløb kan slette tildelingerne for de sidste k dage af den fastlagte periode. Dette betyder, at programmet kun gemmer for eksempel de 12 første dage af en 14-dages periode. Når programmet kører videre og lægger planen for den næste delperiode, bliver den sidste weekend af forrige periode lagt igen. Dette giver en mulighed for at lægge vagterne mere hensigtsmæssigt i denne weekend.



Figur 6.3: Vagtplanen opdeles i flere dele

6.4.3 Opbygning af søgetræ

For at hjælpe søgningen på vej leder vi den i retning af, at nogle af de bløde begrænsinger forsøges opfyldt. Derefter følger vi den fremgangsmåde, vi mødte ude på sygehusafdelingerne med at tildele de vagter, der synes sværest at besætte.

Fridagsønsker

Da hensynet til fridagsønsker er opstillet som en blød begrænsning, vil systemet ikke prøve at opfylde disse i første omgang. Derimod vil der ske det, at en funden løsning kan blive meget dyr, idet det langt fra er sikkert, at fridagsønskerne bliver opfyldt.

Dette retter vi op på ved som det første at forsøge at opfylde fridagsønskerne. Dette stemmer godt overens med praksis, hvor planlæggerne tager store hensyn til personalets fridagsønsker.

Hver sygeplejerske har for hver dag en variabel `SomeHoliday`, der ved hjælp af et heltal angiver, om der ønskes fri eller ej. En værdi større end nul betyder at der ønskes fri med denne værdis prioritet. Hvis der ønskes fri skal solveren forsøge at give medarbejderen fri den pågældende dag (`SomeDay`). Nedenstående kode udtrykker dette med en `if-then-else`-sætning:

```
( SomeHoliday > 0 ->
  try_day_off(SomeDay);
  true
)
```

Sætningen udtrykker at *hvis* `SomeHoliday` er større end nul, *så* prøver vi at sætte `SomeDay` til en fridag, ellers gør vi ingenting.

Vi har programmeret et `try_day_off`-prædikat. Prædikatet udnytter at relationer bliver behandlet i den rækkefølge, de defineres i.

```
try_day_off(X) :- X #= f of shift.
try_day_off(_) :- true.
```

Når `try_day_off(X)` kaldes, testes første tilfælde, og solveren forsøger at sætte variablen `X` til en fridag. Hvis dette ikke går godt, prøves `try_day_off` for andet tilfælde, der altid evaluerer til `true`, og ikke begrænser domænet for `X`.

For at øge effektiviteten af denne søgning har vi konstrueret en overflødig begrænsning, som sørger for at mindske søgetræet.

Som vi har beskrevet tidligere, har vi lavet en hård begrænsning, der sørger for, at antallet af tildelte dag-, aften-, og nattevagter altid stemmer overens med behovet. I praksis betyder dette, at der for hver dag er tre forskellige behov, der skal opfyldes, et for hver vagttype. Disse tre begrænsninger afhænger af hinanden, men dette bliver først opdaget langt nede i søgetræet.

Som eksempel ser vi på en dag, hvor der er et behov for to dagvagter og to aftenvagter samtidig med, at alle har ønsket fri. I første omgang vil begrænsningerne kun sørge for, at der er mindst to, der ikke får fri. Når modellen dernæst forsøger at sætte en af disse to til en aftenvagt, går det galt, da begrænsningen vedrørende dagvagter ikke kan blive overholdt. Modellen vil derfor gå et skridt tilbage i søgetræet og prøve noget andet, som formentlig også vil gå galt. Der er ofte meget lang vej op ad søgetræet til et valg, hvor mindst fire medarbejdere ikke får fri.

For at undgå at en sådan situation opstår, har vi tilføjet en begrænsning, der for hver dag sikrer, at antallet af tildelte fri- og sovevagter stemmer overens med antallet af medarbejdere, der ikke skal bruges på arbejde. Umiddelbart kan denne begrænsning virke overflødig, idet de tre første begrænsninger, netop udtrykker dette. Ved at medtage begrænsningen alligevel, har solveren mulighed for tidligere at stoppe for tildelingen af frivagter og dermed beskære søgetræet betydeligt.

Resultatet er, at søgetræet tidligere får skåret mange undertræer fra, og det omtalte eksempel vil ikke kunne forekomme.

Samlede weekendnattevagter

Efter at have forsøgt at opfylde fridagsønskerne fortsætter vi med at fordele nattevagterne i weekenden. Denne fremgangsmåde er også inspireret af praksis på afdelingerne.

Da vi forsøger at få så få forskellige medarbejdere som muligt på arbejde i weekenden, forsøger vi kun at tildele nattevagter til dem, der skal have 2 eller flere nattevagter i alt. Dette gøres igen ved hjælp af `'->'`-prædikatet. Ligesom i forbindelse med ferieønskerne benytter vi en `try`-funktion. Her prøver `try_nightshift (X,Y)` at tildele både lørdags- og søndagsnattevagten, er dette ikke muligt, tildeles ingen vagter.

```
( OccOfShifts[NurseC,n of shift] >= 2 ->
  try_nightshift(Schedule[NurseC,DayC],
                 Schedule[NurseC,DayC+1]);
  true
)
```

Samlede weekendvagter

Det næste skridt, vi tager, minder om det, som vi netop har foretaget for weekendnattevagterne. For hver medarbejder forsøger vi at tildele en aftenvagt lørdag og en søndag, derefter forsøger vi for hver medarbejder at fordele to dagvagter lørdag og søndag, hvis ingen af delene er mulige, tildeles igen ingen vagter.

Nattevagter

Nattevagterne kan være svære at besætte. En tildeling af en nattevagt medfører flere begrænsninger, end det er tilfældet med de andre vagttyper. Dels har hver medarbejder et fast antal af nattevagter, dels er det meget begrænset, hvad der kan tildeles dagen efter en nattevagt. Derfor er det næste naturlige skridt at sørge for at tildele alle nattevagterne. Igen stemmer denne beslutning godt overens med fremgangsmåden, der benyttes i praksis på afdelingerne.

Vi gennemgår igen alle medarbejderne og benytter et prædikat
`try_night_shift(X)`, som svarer til de ovenfor beskrevne `try`-prædikater.

Opfyldelse af minimumskrav

På nuværende tidspunkt har vi forsøgt at tildele nogle af vagterne. Det næste skridt er at forsøge at fordele vagterne ligeligt.

Vi benytter de tal, vi beregnede vedrørende ligelig fordeling af vagter. Disse tal definerer et interval, som repræsenterer de øvre og nedre grænser for det ønskede antal af henholdsvis dag- og aftenvagter til hver medarbejder. I afsnit 3.1.5 definerede vil disse intervaller til:

$$[\lceil X_{s,a} \rceil - 1; \lceil X_{s,a} \rceil], \tag{3.6}$$

hvor a er dag- eller aftenvagt. Målet er nu at tildele den nedre grænse for antallet af dag-, og aftenvagter til hver medarbejder. Det vil sige, at antallet ikke skal overstiger $\lceil X_{s,a} \rceil - 1$. Således tilstræber vi, at hver medarbejder ikke får flere vagter hver type, end vi har fastlagt, at hun helst skal have.

Metoden til dette foregår ved at se på en medarbejders vagtplan ad gangen og derefter en vagttype ad gangen. Hvis medarbejderen har færre end $\lceil X_{s,a} \rceil - 1$ af vagttype a, fortsætter vi med dennes vagtplan og forsøger at tildele endnu en vagt af type a.

I forbindelse med dette benytter vi et prædikat fra ECL^{i}PS^e :

```
deleteffc(X,Vars,Rest)
```

Prædikat finder ud fra en liste af variable `Vars` den variabel `X`, der har det mindste domæne og som er mest begrænset. Denne variabel fjernes fra listen `Rest`.
I vores program svarer `Vars` i første kald til medarbejderens samlede vagtplan, `X` er en af dagene i vagtplanen, og `Rest` er resten af vagtplanen. Ved igen at benytte et `try`-prædikat forsøger vi at tildele vagttype a på denne dag. Efter dette går vi videre i medarbejderens vagtplan, idet vi ved hjælp af `deleteffc` finder næste variabel. Således bliver vi ved, indtil medarbejderen har fået det ønskede antal af vagttypen a, eller indtil vi har været hele medarbejderens vagtplan igennem.

Resterende vagter

Når vi er kommet hertil, er langt de fleste vagter lagt fast. Der vil derfor ikke være så mange muligheder tilbage for at lægge resten af vagtplanen, hvorfor vi lader systemet om at tildele de resterende

vagter. Dog sørger vi igen for at benytte `deleteffc`-prædikatet. Det vil sige starte med at behandle de variable, der har de mindste domæner og som er mest begrænsede.

6.4.4 Tilfældig rækkefølge

I forbindelse med opbygningen af søgetræet gennemgår vi flere steder mængden af sygeplejersker fra en ende af i fortløbende rækkefølge. Den første sygeplejerske i matricen bliver derved altid behandlet først. Dette medfører en vis form for skæv fordeling, idet den sidste sygeplejerske får tildelt sine vagter senere end alle andre.

For at kompensere for dette har vi indbygget en mulighed for at vælge sygeplejerskerne i tilfældig rækkefølge. Hvis man beslutter at køre programmet med denne mulighed slået til, vil det betyde, at modellen for hvert af de beskrevne punkter vælger sygeplejerskerne i en tilfældig rækkefølge.

Fordelen ved løbende at benytte en tilfældig rækkefølge er, at det ikke altid er den sidste medarbejder i listen, der bliver tilgodeset sidst. Ved at vælge en tilfældig rækkefølge er det tilfældigt, hvem der bliver forfordelt. Ulempen er, at man mister overskueligheden over, hvad der sker i programmet, og at træet blandes undervejs, så man risikerer at komme ned i tidligere afskårne grene, dette kan ikke ske, hvis man benytter et statisk træ.

6.4.5 Opsamling

For at effektivisere søgningen har vi altså forsøgt at opbygge søgetræet således, at de bedste løsninger ligger i venstre side af træet. Vi har placeret følgende beslutninger øverst i træet ved at forsøge at opfylde dem først.

1. Fridagsønsker
2. Samlede weekendnattevagter
3. Samlede weekendvagter
4. Nattevagter
5. Opfyldelse af minimumskrav
6. Resterende vagter

Derudover er der indlagt mulighed for at splitte perioden op i flere delperioder, hvorved søgningen indeholder færre *choicepoints* pr. gennemløb. Desuden er det ved ovenstående punkter muligt at vælge sygeplejerskerne enten i fortløbende eller i tilfældig rækkefølge.

6.5 Mulige udvidelser

I det følgende vil vi kort komme ind på mulighederne for at tilpasse programmet til andre forhold, end dem vi har modelleret.

Vi vil gerne understrege, at følgende kun er overvejelser, og altså ikke er afprøvet i praksis.

Man skal holde for øje, at modellen hurtigt bliver mere kompliceret, og dermed kan den blive meget tungere at løse, end det er tilfældet med den nuværende. Antallet af variable vokser meget hurtigt, da nye begrænsninger ofte medfører mange ekstra variable til hver medarbejder og til hver dag. Derfor bliver det mere og mere vigtigt, at holde øje med at modellen bliver bygget op så hensigtsmæssigt som muligt.

6.5.1 Overførsel fra periode til periode

En logiske udvidelse til programmet er at lave en overførsel fra en periodes vagtplan til den efterfølgende periodes vagtplan. Derved vil det være muligt at skabe en sammenhæng mellem to på hinanden følgende vagtplaner. Dette kan klares ved at lægge en vagtplan for en uge mere end man ønsker ad gangen, hvoraf den første uge er fastlagt af den forrige periodes vagtplan. Dette svarer til den teknik, vi benytter i forbindelse med opsplitning af planen i afsnit 6.4.2.

Hvis dette blev implementeret, ville weekenden op til et periodeskift hænge bedre sammen med den efterfølgende plan. Endvidere kunne man til hver medarbejder knytte en variabel, der fortæller om, hvor mange arbejdsdage han/hun har tilgode i forhold til sit timetal eller de andre medarbejdere. Så er det ikke altid den samme medarbejder, der får en vagt mere end de andre.

6.5.2 Ferie og fridage

Vi har vist muligheden for fridagsønsker. Der er mange andre muligheder. For eksempel kunne man lave variable til at udtrykke ønsker om fri på enkelte vagter, ønsker om fri flere dage i træk eller faste ønsker fra uge til uge. Der er således mange muligheder for at tilgodese medarbejdernes fri-ønsker. Desuden ville det være naturligt at indbygge muligheden for at give medarbejdere ferie, således at man kan tage en medarbejder helt ud af vagtplanen i nogle specificerede dage.

6.5.3 Jul og Nytår

Ønsker man at tilføje en funktion, der er i stand til at fordele jule- og nytårsvagter, vil vi anbefale, at man tager udgangspunkt i den metode, vi mødte på skadestuen i Holbæk beskrevet i bilag A.5. Her skifter prioriteterne fra år til år, og hver medarbejder får på denne måde både år med gode vagter og år med dårlige vagter. Dette forekommer os som den metode, der tilgodeser flest mulige medarbejdere og er mest retfærdig. Teknisk set er det muligt at knytte en prioritet til hver medarbejder og at have en prioriteret liste af vagttyper. For hver tildeling tildeler man så den medarbejder med højeste prioritet, den højeste prioriterede mulige vagt. Det næste spørgsmål er så, om det er praktisk muligt at få medarbejderne til at acceptere, at en computer skal bestemme, hvorledes deres helligdage skal forløbe.

6.5.4 Økonomi

Det skulle ikke være et problem at få den økonomiske side af sagen ind i problemstillingen. For eksempel er det muligt at tilføje endnu en faktor til objektfunktionen. Det kunne eksempelvis være et ønske om at minimere antallet af ansatte sygeplejersker.

6.5.5 Kvalifikationer

Vi har set på muligheden for at angive specielle kvalifikationer i tilknytning til ens embede. Dette kan selvfølgelig udvides til at dække andet end nattevagter, for eksempel ambulancekvalifikationer eller en skelnen mellem sygeplejersker og social- og sundhedsassistenter. Metoden er den samme, som vi har benyttet i vores model, hvor det er muligt at skelne, om hver medarbejder kan eller ikke kan sættes på nattevagt.

Kvalifikationskravene behøver ikke være hårde. De kan gøres bløde ved at lade variablene, der angiver kvalifikationerne (i dette tilfælde n_s), være positive i stedet for binære. Begrænsningen skal så tælle en pris sammen for overtrædelsen i stedet for at umuliggøre den.

6.5.6 Vagter

De faste vagter vi har benyttet (7-15, 15-23, 23-7) er ikke generelle i den forstand, at de passer til enhver afdeling. Dette er der flere muligheder for at rette op på.

Man kan oprette flere typer vagter, for eksempel dag1, dag2, aften1 osv. alt efter hvor mange typer man har. Dette kræver, at man derefter retter programmet til, både med med hensyn til at opdatere antallet af vagttyper og med hensyn til at tilpasse begrænsningerne. Blandt andet skal man på ny definere hvilke vagter, der må efterfølge hinanden. Denne metode gør heller ikke modellen mere generel, men ville være en procedure, man skulle igennem for hver enkelt afdeling. Dette bliver let et større arbejde og umiddelbart vil vi ikke anbefale denne fremgangsmåde.

En anden mulighed er i stedet at ændre på selve modelleringen og til hver vagt foruden en type også knytte et starttidspunkt og et sluttidspunkt. Således knyttes begrænsningerne mere til klokkeslet og tidsrum end til specifikke vagter. Det vil stadig kræve en tilpasning af begrænsningerne, men dette vil være en fælles procedure for mange afdelinger. Derefter skal hver afdeling blot oprette sine vagter.

6.5.7 Ændringer

En fordelagtig udvidelse er at se på mulighederne for at klare behovet ved sygemelding. En mulighed er, at lade det stykke tid der er gået af vagtplanen ligge fast, og derefter lade det koste noget at ændre de andre vagter. På denne måde vil programmet forsøge at minimere antallet af vagter, der ændrer sig. Her vil det være godt med en ide til, hvordan programmet skal gå frem. Man kunne for eksempel lede efter to, der kan bytte vagter.

6.5.8 Bedre evaluering

Det ville være en fordel, at have mere viden om hvornår en vagtrækkefølge og en vagtplan er god, og hvornår den er knap så god

Dette kunne afhjælpes ved at foretage nye samtaler med vagtplanlæggere og sygeplejersker og få en vurdering af, hvilke vagtrækkefølger og hvilke vagtplaner der er bedre end andre. Alt i alt ville dette kunne bidrage til en mere nuanceret objektfunktion.

Kapitel 7

Afprøvning

Vi vil nu sammenligne de to modeller ved at lade dem løse en række testproblemer. Først beskriver vi kort, hvorledes problemerne er opbygget og dernæst hvilke parametre, der er benyttet i afprøvningen. Til slut sammenholder vi resultaterne fra de to metoder.

7.1 Testdata

7.1.1 Testproblemer

Som udgangspunkt er de fleste af vore testproblemer bygget op over de forhold, vi mødte på hospitalsafdelingerne. De konkrete testdata kan aflæses i programfilen `data.ecl` i bilag C.1.5.

De testproblemer, vi benytter, er dels problemer med virkelighedsnære data, dels problemer hvor vi undersøger inddataparametrenes indflydelse. Eksempelvis ser vi på, hvad vil der ske, hvis alle ønsker fri samme dag.

Hillerød

Dataene i de to første tests minder om forholdene hos lægesekretærerne på Hillerød Sygehus.

Her er behovet i hverdagene fastsat til tre dagvagter, fire aftenvagter og to nattevagter. I weekenden falder behovet for aftenvagter til tre.
Der er ansat 18 lægesekretærer, hvilket er det antal, vi har benyttet i `hil18`. Desuden har vi sørget for, at de benyttede timetal stemmer overens med de faktiske.

`hil18`	Test med 18 medarbejdere.
`hil20`	Test med 20 medarbejdere.

Holbæk

De tests, der er navngivet `'hol...'`, tager udgangspunkt i de forhold, der eksisterer på skadestuen i Holbæk. Her arbejder 13 lægesekretærer, som er ansat på stort set ens timetal.

Ud fra basiseksemplet har vi konstrueret forskellige tests til at afprøve de forskellige muligheder i vores program.

`hol13`	Minder meget om de oprindelige forhold på skadestuen i Holbæk.
`hol13t2`	Her er personalet opdelt i to grupper med forskellige timetal. Den ene gruppe har et timetal på 10 og den anden et på 20.
`hol13n2`	Igen er personalet opdelt i to grupper. Denne gang kan den ene gruppe tage nattevagter, mens den anden ikke kan.
`hol13f`	Her tager vi udgangspunkt i `hol13` men tilføjer fridagsønsker for nogle af medarbejderne.
`hol13fa`	I denne test har næsten alle medarbejdere fridagsønsker.
`hol13f1`	Alle medarbejdere ønsker fri på den samme dag.
`hol130`	Et stort eksempel. Denne test er magen til `hol13`, dog har vi multipliceret behovet og personalemængden med 10.
`hol13u8`	Som `hol13` blot for 8 uger.

Andre tests

Her har vi et meget simpelt eksempel, der ikke skulle volde problemer med hensyn til vagtplanlægningen, og et eksempel, hvor der forekommer store forskelle i behovet.

`sim10`	Simpelt tilfælde. Behovet består af en af hver type vagt pr. dag, antallet af medarbejdere er 10.
`var13b`	Her har vi defineret et mandskabsbehov, der svinger voldsomt fra dag til dag.

7.1.2 Valg af søgeparametre

Hvert problem har vi forsøgt at løse både ved hjælp af GAMS og ved hjælp af fire forskellige fremgangsmåder for CP-modellen. Fælles for de fire fremgangsmåder i CP-modellen er parametrene til *branch-and-bound*-prædikatet, hvor vi har valgt følgende:

Strategy: Vi har valgt at benytte `step`-strategien, da denne har givet gode resultater i forbindelse med den løbende afprøvning under udvikling af modellen.

Timeout: Denne har vi sat til 1000 sekunder, hvilket stemmer overens med standardindstillingen i GAMS. Ved de testkørsler, hvor vi har opsplittet vagtplanen i to dele à 14 dage, har vi sat tiden til 500 sekunder for hvert gennemløb, således at summen også her er 1000 sekunder.

From: Da vi ønsker at finde en løsning, der overholder alle de bløde begrænsinger, er den nedre grænse for en løsning sat til 0.

To: Værdien her er sat til 2000, hvilket er en meget høj pris for en funden løsning.

Forskellen på de fire kørsler med CP-programmet er:

Standard: Her lader vi modellen arbejde med alle 4 uger ad gangen og uden brug af muligheden for at se på sygeplejerskerne i tilfældig rækkefølge.

Opsplitning: Vagtplanen splittes op i to delperioder på hver 14 dage. Efter første gennemløb gemmer vi kun på de første 12 dage.

Tilfældig rækkefølge: Her lader vi modellen se på sygeplejerskerne i tilfældig rækkefølge.

Opsplitning + tilfældig rækkefølge: Dette er en kombination af de to foregående fremgangsmåder.

7.1.3 Hardware

Følgende computere er blevet benyttet til vores afprøvninger.

Navn	Model	Ram	Clock	OS
sunfire	SUN Fire 3800	8 GB	750 MHz	Solaris 8
serv2	J7000	4 GB	440 MHz	HP-UX 11i
serv3	J7000	4 GB	440 MHz	HP-UX 11i

Tabel 7.1: Computere brugt til test

'Serv2' og 'serv3' er blevet benyttet til kørslerne i GAMS, og 'sunfire' blev benyttet til afviklingen i ECL^{i}PS^e . Vi har ikke brugt ens computere, da vi ikke havde adgang til computere, hvor både GAMS og ECL^{i}PS^e var installeret.

7.2 Resultater

Resultaterne af testkørslerne er opsummeret i følgende skema, der beskriver, hvor god en løsningen er, og hvor lang tid det tog at finde den.

	GAMS	Standard CP		Opsplitning (14,2)				Tilfældig rækkefølge		Opsplitning + tilfældig			
				14 dage		28 dage				14 dage		28 dage	
Case	pris	pris	tid	pris	tid	pris	tid	pris	tid	pris	tid	pris	tid
hil18	-	-	-	-	-	-	-	-	-	-	-	-	-
hil20	-	**10**	**298**	1	7	-	-	**14**	**5**	8	12	-	-
hol13	-	**4**	**23**	2	1	**2**	**65**	**4**	**6**	5	105	**2**	**5**
hol13t2	-	**5**	**19**	1	4	**14**	**396**	**9**	**8**	2	1	**13**	**6**
hol13n2	**1**	**1**	**28**	0	7	**1**	**22**	**5**	**40**	0	57	**1**	**10**
hol13f	-	**5**	**40**	2	2	**2**	**7**	**8**	**2**	2	2	**2**	**111**
hol13fa	**14**	-	-	2	1	**2**	**5**	**4**	**210**	4	3	**4**	**201**
hol13f1	-	**10**	**321**	-	-	-	-	**7**	**10**	-	-	-	-
hol130	*	*	-	20	22	-	-	*	-	-	-	-	-
var13b	**20**	-	-	-	-	-	-	-	-	-	-	-	-
sim10	**2**	**0**	**10**	0	5	**2**	**3**	**2**	**4**	0	1	**3**	**2**

-: Programmet kunne ikke finde en løsning inden for den fastsatte søgetid.

*: Der er for mange variable og der bliver for mange iterationer
GAMS: `'Iteration limit exceeded'`,
ECL^{i}PS^e : `'Overflow of the global/trail stack'`

Tabel 7.2: Testresultater

Vagtplanerne, der blev dannet ved kørslerne, kan ses i bilag D.

For hver kørsel har vi angivet prisen for den fundne løsning. Desuden har vi for kørslerne i ECL^{i}PS^e angivet, hvor lang tid der gik, før den angivne løsning blev fundet.

Ved de kørsler hvor perioden er opsplittet i to dele, er der angivet to sæt tal. De første svarer til de værdier, der blev fundet efter første gennemløb (14 dage), og de næste to angiver værdierne efter sidste gennemløb. I tidskolonnerne angiver den første værdi tiden, det tog at finde en løsning for de første 14 dage, og den anden værdi angiver tiden, der dernæst blev benyttet til at finde en løsning for 28 dage.

7.2.1 8 uger

Vi har også forsøgt at lade CP-modellen lægge en vagtplan for 8 uger. Ved denne test har vi taget udgangspunkt i `hol13` og blot forlænget perioden. Vagtskemaerne fra disse kørsler kan ligeledes ses

i bilag D. Vi har lavet kørsler uden at splitte perioden op, ved at splitte perioden i to dele à fire-uger, og endelig ved at splitte perioden op i fire dele à to uger. Dette gav følgende resultater:

Periode på 8 uger								
hel periode	pris				tid			
	12				**72**			
2 à 4 uger	pris		tid		pris		tid	
	4		23		**7**		**62**	
4 à 2 uger	pris	tid	pris	tid	pris	tid	pris	tid
	2	1	2	66	0	20	**3**	**124**

Tabel 7.3: Test for 8 uger

Resultatet af disse kørsler er lovende. Ved hver kørsel har ECL^{i}PS^e reelt brugt under to minutter på at finde en god vagtplan. Den kørsel der var længst tid om at finde løsningen, var den, hvor vi delte perioden op i fire dele, men til gengæld har denne løsning den mindste pris.
Umiddelbart ser det ud til, at der ikke er nogen problemer forbundet med at lægge vagtplaner for længere perioder.

7.2.2 Søgetid

Det skal bemærkes, at der er forskel på den computer, som blev benyttet af
ECL^{i}PS^e, og de der blev benyttet af GAMS. Ved at betragte MHz-tallet kan man se, at ECL^{i}PS^e har fået væsentlig mere regnetid end GAMS. Dette har dog ikke stor indflydelse på resultaterne, da de fleste af de løsninger som ECL^{i}PS^e fandt blev fundet efter meget kort tid (under 30 sekunder). Disse løsninger ville derfor også være blevet fundet, hvis ECL^{i}PS^e var blevet kørt på samme maskine som GAMS.

Det eneste tilfælde, hvor over halvdelen af søgetiden blev udnyttet, er `hol13t2`. Løsningen til dette tilfælde var ikke nødvendigvis blevet fundet, hvis en computer som 'Serv2' var blevet benyttet.

Umiddelbart ser vi, at hvis ECL^{i}PS^e finder en løsning, sker dette ofte indenfor kort tid. Dette underbygges også af, at vi også har prøvet at lade ECL^{i}PS^e køre i flere timer, uden at der blev fundet bedre løsninger end de allerede fundne.

7.2.3 Fremgangsmåder

Af skemaet kan man se, at det er forskelligt hvilke fremgangsmåder i søgningen, der giver de bedste resultater.
Ved de tests hvor medarbejderne har forskellige timetal, er der en tendens til, at de fundne løsninger er dyrere og tager længere tid. I langt de fleste tests er det netop prisen for fordelingen af vagter, der giver et bidrag til objektfunktionsværdien.
Dette tyder på, at vores søgesstrategier kan forbedres på det punkt. Dette stemmer godt overens med opbygningen af vores program, der konsekvent prøver at opfylde krav til weekender og fridage for hver gang, der tildeles en vagt. På denne måde bliver disse ønsker implicit vægtet mere end fordelingen af vagter.

I de fleste tilfælde er det en fordel at splitte vagtplansperioden op i flere dele, mens det at behandle medarbejderne i tilfældig rækkefølge giver dyrere løsninger.

7.2.4 Afdelinger med for lidt personale

Ingen af programmerne kunne finde en løsning i tilfældet `hil18`, der er en afdeling med meget overarbejde. Derudover har afdelingen også lokalaftaler, der gør det muligt at nedsætte hviletiden. Her kan vores skarpe krav om 11 timers hviletid og kun en vagt pr. dag meget vel tænkes at være en stor hæmsko for løsningen.

`hil20` er fremkommet ved at bløde `hil18` lidt op ved at tilføje flere medarbejdere. Her er det muligt for ECL^{i}PS^e at finde en løsning, mens det ikke lykkedes for GAMS.

7.2.5 Varierende behov

I `var13b` svinger behovet meget fra dag til dag og uge til uge. I dette tilfælde finder GAMS en løsning med en omkostning på 20, mens det ikke lykkedes for ECL^{i}PS^e at finde en løsning. I forhold til mandskabsplanlægning på sygehuse er dette dog ikke af særlig betydning, da behovet i praksis ikke varierer så voldsomt som i eksemplet.

7.2.6 Fridagsønsker

I de tests der indeholder fridagsønsker, ser vi, at disse stort set altid bliver overholdt. Dette er som forventet, da det netop er fridagsønskerne, som CP-modellen forsøger at overholde først blandt de bløde begrænsninger.

7.2.7 Generelt

Overordnet er tendensen den, at vores CP-program finder væsentlig bedre løsninger end GAMS-programmet. Desuden bliver løsningerne også fundet hurtigere ved hjælp af CP.

Et af problemerne for vores program er at fordele vagterne ligeligt. Dette kunne afhjælpes ved at lægge antallet af vagter fast for hver medarbejder fra start af. Et sådant valg ville medføre, at alle andre krav ville blive sværere at opfylde, når man skal finde en løsning. Især de faste krav såsom friweekender og vagtrækkefølger ville blive svære at opfylde, da de inkluderer bestemmelser for flere dage ad gangen.

Vi så, at ECL^{i}PS^e fandt en rimelig god løsning på kort tid og derefter ingen bedre løsninger indenfor den fastsatte søgetid. Dette tyder på, at vores søgetræer opbygges fornuftigt, idet der hurtigt findes en god løsning.

Dog bemærkede vi, at forskellige strategier gav forskellige løsningsværdier. Dette tyder på, at søgningen alligevel kan forbedres.

Ovenstående to punkter tyder også på, at gode løsninger i søgetræet ligger langt fra hinanden.

Alt i alt understøtter vores afprøvninger, teorien om at det er vigtigere at have gode søgerutiner og fornuftig træ-opbygning end at have lange søgetider.

Kapitel 8

Konklusion

Vi har udviklet et CP-program, der kan finde løsninger til vagtplanlægningsproblemer indenfor kort tid. Endvidere har vi opnået indsigt i problematikken omkring vagtplanlægning på sygehuse.

Vagtplanlægning på sygehuse er et problem med mange begrænsninger. Vi har set problemet løst på forskellige måder. Fælles for dem alle er, at vagtplanen lægges i hånden, og at det tager lang tid at lægge en vagtplan. Planlæggerne på sygehusafdelingerne har stor erfaring med at lægge vagtplaner og stor heuristisk viden om, hvilke beslutninger der kan lade sig gøre, og hvilke der ikke kan.

Rent modelleringsmæssigt er der mange krav, der skal overholdes, og problemet er meget kompliceret, i og med at fastlæggelse af en variabel har stor indflydelse på mulighederne for de resterende variable. Uanset hvilken fremgangsmåde, der benyttes, bliver søgetræet hurtigt meget stort. Et valg, der foretages tidligt i søgningen, kan have stor konsekvens til slut, men det er svært at forudsige, hvilke valg der er gode, og hvilke der er dårlige.

Under udviklingen af de to modeller viste det sig, at der var en del aspekter, der var lettere at udtrykke i constraint programming-modellen end i den lineære model.

Den matematiske model blev hurtig meget stor og uoverskuelig. Derudover modelleres diskrete krav forholdsvis klodset ved hjælp af binære variable, og man skal hele tiden holde for øje, at problemet skal lineariseres.

CP kræver ikke, at man lineariserer udtrykkene, da disse blot testes undervejs i kørslen. Endvidere er diskrete krav lette at modellere direkte ved hjælp af logiske disjunktioner. Variable med heltalsdomæner giver ingen problemer i form af relaxering. Dermed åbner CP muligheden for at modellere mere virkelighedsnære problemer.

Man skal dog være mere forsigtig under modelleringen i CP, end man skal i lineær programmering. Dette gælder både med hensyn til programterminering og med hensyn til, at der kun sker variabeltildelinger der, hvor det rent faktisk er meningen.

Regelrækkefølgen har stor indflydelse på søgetiden i CP. Hvis man tilføjer en ekstra regel til programmet, kan det godt betale sig at tænke nøje efter, hvor den skal indsættes. I GAMS har rækkefølgen ingen betydning.

Med hensyn til køretid halter CP lidt bagefter, hvis man blot benytter den indbyggede `labeling`. Her er den indbyggede solver i GAMS klart mere effektiv. Til gengæld er det forholdsvis let selv at angive søgestrategier i CP, og der skal ikke meget til, før effektiviteten forbedres betydeligt.

Spørgsmålet om hvorvidt man vil have en god løsning eller en optimal løsning er altafgørende i CP. Ved hjælp af søgeprocedurer og fornuftig programopbygning er det ofte muligt hurtigt at finde gode bud på en løsning. Derimod kan det godt være yderst ressourcekrævende og til tider for ressourcekrævende at finde den bedste løsning.

For at løse problemer ved hjælp af CP er det langt fra nok blot at modellere disse. Hvis man skal finde en god løsning, og helst inden for rimelig tid, er det nødvendigt at formulere en egentlig søgestrategi. Dette gør CP-programmer mere problemspecifikke.

8.1 Opsamling

Alt i alt mener vi, at vi har opnået en god viden indenfor vagptlanlægningsproblemer og har fået stor indsigt i CP-paradigmet.
Vi mener, at CP-paradigmet er velegnet til løsning af vagtplanlægningsproblemer. Dette bygger vi på, at vi har fundet gode løsninger på vagtplanlægningsproblemer på forholdsvis kort søgetid.
Grunden til, at vi finder CP velegnet til vagtplanlægningsproblemer, er, at vi er tilfredse med en god løsning fremfor den bedste løsning. I realiteten er en løsning med en lille omkostning altså acceptabel. Sammenholdes dette med fremgangsmåden ude på afdelingerne i praksis, er der her heller ingen garanti for, at den bedste løsning findes, da det for planlæggerne er umuligt at overskue alle muligheder for at lægge en vagtplan.

Vi finder, at CP er et spændende og kompetent udviklingsværktøj i forbindelse med vagtplanlægning. Der er gode muligheder for at modellere krav og ønsker til vagtplaner. Desuden er der gode muligheder for at udvikle forskellige søgemetoder og modeller.

Vi har fået udviklet en basismodel, der kan finde løsninger til vagtplanlægningsproblemet med forholdsvis små priser, og vi mener, at der er gode muligheder for at udvide modellen, så den passer til andre mandskabsplanlægningsproblemer - såvel specifikke som generelle.

Bilag A

Mødereferater

I det følgende har vi gengivet vores referater fra hvert af de fem interviews vi har haft med personale fra Hillerød Sygehus, Gentofte Sygehus og Holbæk Sygehus.
I afsnit 2 har vi lavet en samlet oversigt, der beskriver det samlede resultat.

A.1 Spørgsmål benyttet til interviews

Ved hvert interview benyttede vi os af følgende spørgsmål, som alt i alt dækker den mængde af information, vi mente var nødvendige for at give os indblik i de behov der er til vagtplanlægningen.

Generelle spørgsmål

1. Hvem (og hvor mange) lægger vagtplanen i dag?
2. Hvor lang tid bruger I på at lægge en vagtplan?
3. Hvor lang tid før lægges planen?
4. Hvordan er tilfredsheden med vagtplanen og hvordan måles den?
5. Hvordan fortæller medarbejderne om deres ønsker?

Planlægningen

1. Hvor lang periode planlægges der for?
2. Hvor mange personer planlægges der for?
3. Hvilke vagttyper har I?
4. Hvor mange arbejdstimer har de forskellige medarbejdere? (fuldtid, deltid, andet?)
5. Hvordan er vagterne delt op? (er døgnet delt i tre, eller er der overlap, eller ses der pr time?)
6. Hvordan finder I ud af hvor meget personale der skal bruges på vagt?
7. Er der et ens bemandingsbehov fra dag til dag, uge til uge?
8. Bliver bemandingsbehovet altid dækket?
9. Sker det at der er flere på arbejde end mindstebehovet?
10. Er der nogle hviletidsbestemmelser (11-timersreglen)?
11. Er der et max for antallet af timer man må være på arbejde?
12. Overholdes timeantallet for hver medarbejder pr uge? (eller overholdes gennemsnittet over fx 4 uger.)
13. Har alle medarbejdere samme kvalifikationer, eller er der nogle”typer” der altid skal være på arbejde?
14. Er der nogle vikarer? (fx ved sygdom eller som faste timer?)

Overvejelser

1. Ser I tilbage på den sidste plan, når I laver en ny plan? (for at få et sammenhængende forløb fra plan til plan.)
2. Hvordan klares ønsker om fridage? (timer, halve dage.)
3. Er der faste ønsker der går igen fra uge til uge?
4. Hvor meget tages der hensyn til disse ønsker?
5. Prøver I at samle eller sprede aften- og nattevagterne? (er det evt. individuelt?)
6. Hvad gør I hvis der er for mange der vil have fri samtidig? (fx ved jul og påske.)
7. Tages der hensyn til at nogle arbejder godt/dårligt sammen?
8. Tages der personlige hensyn? (fx mandag morgen, børn, nattevagt.)
9. Hvordan med nattevagter? (ønsker, krav.)
10. Hvordan med fridage? (to dage i træk, en weekend-dag, (f,a,f), ...?)
11. Bliver pauserne inddraget i planlægningen (frokost, kaffepause,...?)
12. Eksisterer der generelle regler (fx alle skal have weekendvagter og nattevagter.)
13. Bliver alle regler overholdt?

Andet

1. Hvilke ønsker har medarbejderne typisk?
2. Bliver vagtplanen løbende ændret? (hvor holdbare er de?)
3. Hvad gør I hvis nogle bliver syge?
4. Hvordan klares behovet i forbindelse med ferier?
5. Er der andet der tages hensyn til i planlægningen?
6. Er der noget der ville være rart at få med i planlægningsfasen?

Data

1. Er det muligt at se nogle gamle vagtplaner, samt medarbejdernes ønsker?
2. Er det muligt at se nogle gamle “behovsplaner”?
3. Er det muligt at indsamle kommende data til os?

A.2 Lægesekretær på ortpædkirurgisk afdeling på Hillerød Sygehus

Møde med Cecilie Kabell, ortopædkirugisk lægesekretær, på Hillerød Sygehus den 30. april 2002.

A.2.1 Mandskabsbehov

Mandskabsbehovet hos lægesekretærerne er fastlagt således at hverdagene alle følger samme dagskema mens weekender og helligdage følger et andet dagskema.
For nattevagterne gælder det at natten mellem fredag og lørdag tæller som en weekendvagt og nattevagten mellem søndag og mandag behandles som en hverdagsvagt. Mandskabsbehovet er opstillet i tabel A.1.

	Dag	Aften	Nat
Hverdage	1 x 7-15 1 x 8-16 1 x 9-15	2 x 15-23 2 x 16-23	1 x 23-06 1 x 23-07
Weekend og helligdage	1 x 7-15 1 x 8-16 1 x 11-19	1 x 15-23 2 x 16-23	1 x 23-06 1 x 23-07

Tabel A.1: Mandskabsbehov, Hillerød Sygehus

A.2.2 Krav til planlægningen

Der er opstillet følgende krav til planlængningen, som skal være opfyldt for alle medarbejdere:

- 2 sammenhængende planlagte fridage for hver 14 dage.
- 2 fridage per uge.
- Arbejde hver anden weekend.
- 8 timers pause efter hver vagt.

I stedet for 11-timersreglen, har de på denne afdeling en regel der siger at de kan nøjes med 8 timers pause efter hver vagt. I praksis er det dog meget sjældent at der er under 16 timer mellem to på hinanden følgende vagter.

Udover de faste regler, forsøges der at tage en række hensyn, som også gælder for alle medarbejderne:

Sammenkædning af weekendvagter Hvis man har en type vagt i starten af weekenden, så vil man oftest have den samme type vagt den efterfølgende dag.
Dette hensyn tages for at opfylde et ønske om at "ødelægge" weekenden for så få medarbejdere som muligt ad gangen.

Rækkefølge ved vagttypeskift Den ideelle rækkefølge af vagterne, hvis disse skifter, er dagvagt, aftenvagt, nattevagt.
For så vidt muligt prøver man at undgå at lægge en dagvagt efter en aftenvagt.
Efter en nattevagt kan man have en dag fri eller også kan man eventuelt have en aftenvagt. Bortset fra i weekenden, har man som regel kun en nattevagt ad gangen.

Til sidst forsøges der at tage så mange personlige hensyn som muligt. Ud over individuelle ønsker, såsom at man gerne vil have fri onsdag aften pga. fritidsaktiviteter el. lign, drejer det sig om:

- Ønske om ekstra nattearbejde (oftes unge).
- Holde alle over 55 år fri for nattearbejde.

- Dele nattevagter og weekendvagter ligeligt ud til alle dem der ikke ellers får personlige hensyn mht. nattearbejde.
- Dele jule- og nytårsvagter ligeligt ud til alle, set over en årrække.

I praksis sørges der for at gå mere efter at følge de forskellige medarbejderønsker, i stedet for at overholde de fastsatte ansættelsesaftaler og fagforeningsregler.

A.2.3 Fremgangsmåde for vagtplanlægning

I øjeblikket er to af medarbejderne ansat til at lægge vagtplanen. For at lægge en vagtplan, som strækker sig over fire uger, bruger de 6 timer hver. Planen skal være lagt mindst 4 uger i forvejen.
For at opretholde en løbende sammenhæng mellem to på hinanden følgende vagtplaner, sørger planlæggerne for at kigge tilbage på de sidste 2-3 date i den foregående plan, når de lægger en ny plan.

I denne afdeling, er der ansat 18 medarbejdere, som alle er ansat på vidt forskellige timetal, lige fra 15 timer pr. uger, til 37 timer pr. uge, til gengæld har alle de samme kvalifikationer. Dette betyder at planen ikke behøver at tage højde for specielle færdigheder hos nogle af medarbejderne.

De ansattes timetal bliver cirka overholdt over en 2-ugersperiode. Dette skal sammenholdes med at hver arbejdstime mellem 17 og 06, svarer til 1+3/37 normtime. I dette tidsrum er arbejdstimerne altså mere værd end ellers.

Samlet set, er der behov for flere arbejdstimer, end summen af de 18 medarbejderes arbejdstider. For at klare dette, bliver de overskydende arbejdstimer vist til alle medarbejderne, der efterfølgende melder sig til de vagter de har mulighed for at tage. Dette betyder at de fleste, i sidste ende, kommer til at arbejde mere end det antal timer de er ansat til. Dette "merarbejde" ordnes ved at medarbejderne får udbetalt ekstra løn, svarende til deres almindelige timeløn - der er altså ikke tale om hverken overbetaling, flekstid eller tilsvarende. Flekstid ville desuden betyde at "hullerne" blot ville blive "skubbet" fremad i tiden.

Rent praktisk foregår skemalægningen ved at indsætte mandskab for de ikke-populære vagter (nattevagter og weekendvagter) først. Derefter startes der fra en ende af, med at udfylde vagtplanerne for en medarbejder ad gangen - i sidste ende bliver der naturligvis behov for at rokere lidt rundt på allerede behandlede medarbejdere.

Som allerede beskrevet, vil vagtplanen slutte med en række huller. Disse huller er som regel hverdagsdag-, eller hverdags-aftenvagter, da disse typer er lettere at afsætte end både weekendvagterne og nattevagterne.
I de fleste tilfælde bliver alle hullerne fyldt ud ved at medarbejderne selv melder sig til disse vagter. Når det en sjælden gang imellem sker at der er en vagt som ikke er blevet taget, prøver man at overtale/presse folk til at tage de sidste ubesatte vagter. Hvis det ikke lykkedes, bliver man nødt til at rokere lidt rundt på de planlagte vagter, således at alle vagter i sidste ende er besat.

A.2.4 Ønsker

Det eneste ønske som Cecilie umiddelbart har, og som ikke er opfyldt i dag, er muligheden for at lægge en rullende 12-ugersplan. Altså en 12-ugersplan som skal kunne bruges igen og igen. Derved ville alle medarbejderne kunne se langt frem i tiden, og vide hvornår de skal arbejde. Desuden ville planlæggerne skulle bruge meget mindre tid, da planen ville være rimellig statisk og kun indeholde småændringer fra gang til gang.

A.3 Overlæge på anæstesiafdelingen på Holbæk Sygehus

Møde med Ole Rudkjøbing, overlæge på anæstesiafdelingen på Holbæk Sygehus den 7. maj 2002.

A.3.1 Baggrund

Ole Rudkjøbing arbejder ikke selv med at lægge vagtplaner, men han har arbejdet en del år på at udvikle et PC-program (Rammeplan), som kan hjælpe med at lægge vagtskemaer og med at holde styr på timeregnskaber.
Da han netop selv har forsøgt at løse et vagtplanlægningsproblem for vagtskemaer på sygehuse, var han naturligvis interessant at tale med for os.

Ud over at høre lidt om hans PC-program, fik vi et indblik i hvilke regler, begrænsninger og ønsker der findes i forbindelse med planlægningen af et vagtskema i hans afdeling. Dog var det ikke muligt at få et ordentligt billede af hvordan planlægningen foregår i afdelingen, da det ikke er Ole Rudkjøbing der lægger vagtplanerne.

A.3.2 Rammeplan

Ole Rudkjøbings planlægningsprogram, er et databasebaseret dos-program, der både kan bruges til at lægge en vagtplan og til at taste en vagtplan ind i og derved holde styr på arbejdstimer og løn.

Når programmet skal lægge en vagtplan, prøver det at lægge en færdig vagtplan uden huller, men det tilgodeser samtidig altid alle medarbejdernes ønsker. Når der opstå konflikter, fx ved at alle vil have fri samme dag, får de i praksis det og programmet undgår at besætte vagten (det kommer altså et hul i vagtplanen). Det betyder altså, at planlæggeren efterfølgende skal løse sådanne konflikter manuelt.

Programmet kigger løbende bagud og bruger et retfærdighedsindeks for hver medarbejder, for derved at få fordelt de svære vagter (natte- og weekendvagter) ligeligt.

A.3.3 Mandskabsbehov

Mandskabsbehovet for lægerne på anæstesiafdelingen, følger samme struktur fra uge til uge. Der eksisterer følgende typer af vagter:

Dagvagt (8-15). Dette er en "almidelig" vagt. Her er lægerne enten normalt på arbejde, eller udfører en ud af de 6 "specialopgaver" som afdelingen også varetager. Disse opgaver er:
- Ambulancevagt - 1 om dagen (varer til klokken 16).
- Udstationering til Nykøbing Sjælland - 1 om dagen.
- Øreafdelingen - 1 om dagen.
- Operationsgang - 3 om dagen.
- Intensiv - 2 om dagen.
- Smerteklinik - 1 om dagen, dog kun 2 gange pr uge.

Bagvagt (8-23). Ved denne vagt, har man først en almindelig dagvagt fra 8 til 15. Derefter har man tilkaldevagt til klokken 23, så man kan kaldes ind hvis der skulle opstå problemer.

Forvagt (15-8). Ved en forvagt møder lægen klokken 15 og går først hjem klokkken 8 næste morgen.

I weekenderne er behovet dog ikke nær så stort.

Både dagen efter en bagvagt og efter en forvagt, har man fri. Den eneste vagt, hvor man skal tage hensyn til specielle kvalifikationer er ambulancevagten, som 5 af lægerne har den nødvendige uddannelse til.

A.3.4 Krav til planlægningen

I denne afdeling er der ikke blevet opstillet specielle faste krav til planen. Dette betyder at det eneste faste krav der skal opfyldes, er de lovmæssige bl.a. 11-timersreglen.
Derudover kommer der naturligvis en række punkter som de ønsker opfyldt. Disse punkter er:

Sammenkædning af weekendvagter. Hvis man har en vagt om lørdagen, vil man oftest også få en vagt om søndagen. Derved vil lægerne få "ødelagt så få weekend som muligt".
Hvis man har haft en weekend-døgnvagt vil man oftest have fri om mandagen.

Ønsker om fridage. De forsøger altid at tilgodese medarbejdernes ønsker om fridage. I realiteten er det meget sjældent man ikke får fri på en ønsket fridag.

Deling af weekend- og nattevagter. Disse vagter bliver spredt ud på alle medarbejderne, således at hvis man ser over en længere periode, vil alle medarbejdere have et lige antal af disse vagter.

A.3.5 Fremgangsmåde for vagtplanlægning

I øjeblikket er to af medarbejderne ansat til at lægge vagtplanen. Den ene bruger ca. en dag på at lægge en grundplan, hvorefter den anden fungere som kontrollant.
Kontrollantens opgave, er at gennemse grundplanen for at sikre at alle medarbejdernes vagtplaner, ser ordentlige ud. Idéen er at det formentlig er lettere for et par "friske" øjne at finde evt. fejl eller uhænsigtsmæssigheder i planen.

En vagtplan strækker sig fra den første i måneden og måneden ud og den skal være lagt mindst fire uger før. I praksis fungerer det dog således at grundplanen er færdig fire uger inden den tages i brug, hvorefter den bliver rettet til, hvis der skulle være nogle nye ønsker fra medarbejderne.

Ved hver plan, sørger planlæggeren for at kigge tilbage på den foregående plan, for dermed at opretholde en løbende sammenhæng mellem to på hinanden følgende vagtplaner.

A.4 Afdelinssygeplesjerske på afdeling 734 på Amtssygehuset i Gentofte

Møde med afdelingssygeplejerske Edel Sonne, på Amtssygehuset i Gentofte Sygehus den 28. maj 2002.

A.4.1 Mandskabsbehov

Mandskabsbehovet hos afdeling 734 er fastlagt efter et fast 3-skifts-ugeskema, hvori der kun skelnes mellem hverdage og weekender. Den eneste forskel mellem hverdage og weekender, er den at mandskabsbehovet er sat en anelse ned for dagvagterne i weekenden. I tabel A.2 ses en oversigt over mandskabsbehovet.

Vagttype	Tidsinterval	Kommentarer
Dagvagt, hverdage	6 x 7.30 - 15.30	Minimum 4 sygeplejersker
Afdelings-sygeplejersken	7.30 - 15.30	En dag kun til kl 12:30
Sekretæren	7.00 - 13.00	
Dagvagt, weekender	4 x 7.30 - 15.30	Minimum 2 sygeplejersker
Aftenvagt	3 x 15.30 - 23.30	Minimum 2 sygeplejersker
Nattevagt	2 x 23.30 - 7.30	Minimum 1 sygeplejerske

Tabel A.2: Mandskabsbehov, Amtssygehuset i Gentofte

Medarbejderne på afdelingen kan deles ind i fire grupper:

1 : Sygeplejersker, 13 stk.
2 : Social- og sundhedsassistenter (Sosuassistenter), 7 stk.
3 : Afdelingssygeplejerske, 1 stk. (Edel Sonne)
4 : Afdelingens sektretær, 1 stk.

Som det ses i skemaet, er de to specielvagter, “Afdelings-sygeplejerske” og “Sektretær” lagt fuldstændig fast, idet der netop er en person ansat til hver af disse vagter. Det interessante i denne sammenhæng er de resterende vagter, som skal deles mellem de 13 sygeplejersker og de 7 sosusassistenter.
Til hvert tidsrum er der fastsat et minimumsantal, for hvor mange sygeplejersker der skal være på vagt. De resterende vagter kan enten blive besat af sygeplejersker, eller af sosusassistenter.

A.4.2 Krav til planlægningen

For det første skal vagtplanerne følge overenskomsten.

Udover overenskomstreglerne, forsøges der at tage så mange individuelle hensyn som muligt. Edel fortalte at hun har lært alle medarbejdernes ønsker, således at hun har fuldstændig styr på hvilken struktur den enkelte medarbejder helst vil have i sit vagtskema. Hun vidste fx. hvem der helst vil have en fridag, inden de skal starte på en række nattevagter, samt hvem der derimod helst ville afslutte en række nattevagter med en ekstra fridag og lignende.

Hvis nogen medarbejdere har ønske om fridage, og hvis de fortæller om dette inden vagtplanen bliver lagt, vil disse ønsker stort set altid blive tilgodeset.
Hvis man derimod har ønske om en fridag, efter vagtplanen er blevet lagt, bliver man nødt til at finde en kollega man kan bytte vagt med.

I praksis sørges der for at gå mere efter at følge de forskellige medarbejderønsker, i stedet for at overholde de fastsatte ansættelsesaftaler og fagforeningsregler.

A.4.3 Fremgangsmåde for vagtplanlægning

Edel Sonne har lagt vagtplanerne for afdelingen i over to år. Dette har naturligvis givet hende en del rutine i, hvordan man griber problemstillingen an. Hun har fået så meget rutine, at hun nu bruger under en dag på at lægge en vagtplan, som strækker sig over 4 til 8 uger.

Edel fortalte at det primære formål med at lægge en vagtplan, ikke var at gøre medarbejderne tilfredse, men derimod at sikre de bedst mulige forhold for patienterne. Hun viste os en oversigt, som beskriver deres afdelings indstilling til det at lægge en vagtplan og som desuden giver et fint billede af hvilke ting Edel prøver at opnå ved hver vagtplan (se tabel A.3.

Fomål	Sikre en kvalificeret pleje af patienterne med bedst mulig brug af personaleressourcerne
Mål	-Kontinuitet fra dag til dag over døgnet -Jævn fordeling af erfarne og nye medarbejdere -Jævn fordeling af sygeplejersker og sosuassistenter -Overholde overenskomster og aftaler -Fordeling af ferier jævnt over året -Kurser -Afspadsering -Personaleønsker

Tabel A.3: Tjenestetidsplanlægning

Ligesom hos vagtsekretærerne i Hillerød, lægger Edel vagtplanen ved at starte med de mest avancerede vagter. Dvs. hun først sørger for at indsætte personale til nattevagter, samt på de dage hvor der fx er nogle der skal på kursus eller lign.
Når hun er færdig med vagtplanen, vil der tit være nogle få huller, hvilket betyder at hun altså mangler personale til disse vagter.

For at få udfyldt hullerne i vagtplanen, prøver hun først at få nogle af de ansatte til at tage en ekstra vagt eller flere - her udnytter hun igen at hun kender personalet, dvs. at hun starter med at spørge dem, som hun ved godt kunne bruge lidt ekstra penge den relevante måned. Det kan fx. være at der er en der lige har haft bilen til reparation, eller en der gerne vil på skiferie.
Hvis det ikke lykkedes at få nogen af de ansatte til at udfylde hullerne i planen, bliver de nødt til at ansætte en fra et vikarbureau. Dette er dog kun en nødløsning, da vikarer dels er en dårlig arbejdskraft (da de ikke kender til afdelingens rutiner osv), dels er det en relativt dyr arbejdskraft.

De medarbejdere der er ansat i afdelingen, er alle ansat på individuelle antal timer, dog er de fleste ansat på fuldtid, altså 37 timer om ugen. De ansattes timetal stemmer altid overens med den gennemsnitlige arbejdstid pr uge fordelt over hver vagtplan, det vil altså sige, at deres arbejdstimer bliver fordelt over en periode på 4 til 8 uger.
Mht. ferie, har de en ordning på denne afdeling, der siger at der altid skal være en sygeplejerske og en sosuassistent der har fri. I sommerperioden ændres dette dog til at der er to af hver der har fri.

A.5 Lægesekretær på skadestuen på Holbæk sygehus

Møde med Pia Varkil, lægesekretær på skadestuen på Holbæk sygehus den 4. juni 2002.

A.5.1 Planlægningsfasen

I denne afdeling er der 1 person (Pia Varkil), der sidder med hele planlægningen. Hun bruger ca. en halv dag for hver plan.

Planen lægges, så den er klar 1 måned før den træder i kraft. Hvis man har nogle ønsker om specielle fridage i den periode den gælder for, bliver der for så vidt muligt taget hensyn til det i planlægningen. Kommer ønsket først efter at planen er lagt, er det ens eget problem og man må finde en af sine kolleger og bytte vagter med.

Eftersom der er de samme bemandignsbehov fra uge til uge, er der gode muligheder for at genbruge noget af planen. Faktisk bruger de her en 4-ugersrulleplan, der kører for et ¡ år ad gangen. Det tager lang tid i starten at få lagt rulleplanen, her bruger Pia cirka 2 halve arbejdsdage. Når den er lagt er det mest småændringer der bliver lavet fra måned til måned.
Hvert halve år lægges rulleplanen om, eller byttes rundt og hver medarbejder kan så aflevere deres ønsker for rulleplanen det næste halve år til vagtplanlæggeren. Dette gøres på en seddel hvor man kan skrive de vagter man godt kunne tænkte sig. Et eksempel på dette kan ses på billede A.6.5. Eftersom alle ikke altid kan få opfyldt deres ønsker, køres der med en prioriteret ønskeliste, hvor man skiftes til at være førstevælger. Dette system gælder også ved ferielægning og ved jul og nytår.

Jul og nytår er specielle og tages ud af rulleplanen. Her skal man arbejde jul (24., 25.) hver andet år og nytår (31., 1.) hvert andet år. Her er det igen efter en ordning hvor man skiftes til at være førstevælger og man kan så vælge hvilke 2 vagter (der skal være af samme type) inden for den gruppe man hører i, man helst vil have. Dette gøres på et skema der kan ses på billede A.6.5.
Sommerferieønsker klares på samme måde, med at man får en prioritet (1-5) tildelt og den der har 1. prioritet får taget mest hensyn til sine ferieønsker. Ferierullesedlen kan ses på billede A.6.5 Et eksempel på hvordan et typisk vagtrul kan se ud på A.6.5.

A.5.2 Mandskabsbehov

Lægesekretærerne på skadestuen og den akutte indlæggelse har nogle faste vagter, som skal være dækket for hvert døgn. Disse vagter er ens fra dag til dag for alle dage også weekender.

Hele afdelingen råder over 14 sekretærer, 2 af dem er ansat til 32 timer pr. uge, mens resten er ansat til 28 timer pr. uge. for hver døgn er der 7 fremmødte. 3 til dagvagt, 3 til aftenvagt og 1 til nattevagt. Dette er illustreret i skema A.4.

Dag, D1	1 x 7-15
Dag, D2	2 x 8-16
Aften, A1	1 x 15-23
Aften, A2	2 x 16-00
Nat, N	1 x 23-07

Tabel A.4: Mandskabsbehov, lægesekretærer på Holbæk Sygehus

Det forholder sig lidt anderledes med den ledende lægesekretær, der møder 8-15:30 i stedet (undtagen 1 dag om ugen hvor det er 8-15:00).

A.5.3 Krav til planlægningen

I planlægningen tages der hensyn til at alle skal arbejde hver anden weekend.

Der er opstillet følgende krav til planlængningen, som i følge overenskomsten, skal være opfyldt for alle medarbejdere (i praksis sørger man dog for at opfylde medarbejdernes ønsker, frem for overenskomstsreglerne):

- For hver 7. døgn skal man have 1 fridag.
- Fri hver anden weekend.
- Arbejde hver anden weekend.
- 4 fridage på 14 dage.
- Efter enkelt nattevagt, har man 1 sovedag.
- Et fridøgn er på 35 timer (evt. 32 på dispensation).
- 11 timers pause efter hver vagt (11-timersreglen).

Udover de faste regler, forsøges der at tage en række hensyn, som også gælder for alle medarbejderne:

Samarbejdsproblemer Hvis to medarbejdere ikke kommer godt ud af det sammen, prøver man at undgå at sætte dem fast sammen. Det kan godt hænde at de sidder sammen en enkelt dag eller to.

Aften-Dagvagt Man prøver at undgå at lægge en dagvagt lige efter en aftenvagt, da dette vil være temmelig hårdt for medarbejderne.

Specielle ønsker Det kan være at en gerne vil have fri hver torsdag aften pga. skolegang eller andet. Dette prøves der for så vidt muligt at tage hensyn til.

Timetal Medarbejdernes timetal forsøges overholdt over 4 uger.

A.5.4 Andet

Når man har arbejdet 37 timer mellem 17 og 6, får man 2 timer ekstra. Denne afdeling er heldig i forhold til mange af de andre afdelinger vi har snakket med, fordi de har en fast natsygeplejerske, på denne måde får de dækket halvdelen af deres nattevagter. Alle de ansatte skal kunne tage en nattevagt, og de kan klare nattevagtfordelingen på frivillig basis.

Ved ferie og sygdom går den ledende vagtsekretær ind og kigger på hvem der bedst kan og hvis tur der er. I disse tilfælde kan ønsker og lign. godt blive tilsidesat.

Rulleplanen dækker alle vagtbehovene, men ferier- og omsorgsdage er ikke medregnet. Dette betyder at der i praksis godt kan ske det at der alligevel er ubesatte vagter.

De har vikarer de kan kalde ind, det kan også ske at der er en for lidt på arbejde på en dag eller aftenvagt. Det er yderst sjældent at der er 2 for lidt på arbejde.

Eftersom jul, og nytår er lidt mere specielle tages disse ud ad rulleplanen og lægges for sig.

Man behøver ikke at tænke på hvilken sygeplejerske man sætter til hvad, da alle kan det samme. Dog tages der hensyn til at nye ikke skal sidde alene, men først skal læres op.

Hvis man får omlagte timer (f.eks. dag vagt skiftes til aftenvagt) så får man et godtgørelsesbeløb.

Har man haft et hårdt halvt år på rulleplanen tages der hensyn til dette i næste rulleplan og man kan så eks. få fri (fredag, lørdag, søndag) dog vil personlige ønsker altid gå forud.

A.5.5 Ønsker

Det er afdelingens ønske at det kunne være lækkert med et program der kunne lægge en færdig vagtplan, hvor eventuelle konflikter også er løste. Den skal altså ved konflikter kunne finde ud af at tage stilling til og afbalencere, hvis tur det er til at arbejde/få fri, samt hvem der bedst kan tage en ekstravagt i forhold til sit arbedsskema.

A.6 Afdelingssygeplejerske på afdeling A2 på Holbæk Sygehus

Møde med Lisbet Pedersen, afdelingssygeplejerske på ortopædkirurgisk afdeling på Holbæk Sygehus den 4. juni 2002.

Lisbet Pedersen står for vagtplanlægningen for i alt 50 sygeplejersker. Dette betyder at hun arbejder med at langt større mandskab end de andre afdelinger vi har været ude at se på. Hun er fuldtidsansat og bruger cirka halvdelen af sin tid på at lægge vagtplaner.

A.6.1 Mandskabsbehov

Behovet på denne afdelingen er ikke fast, som vi har set det på andre afdelinger. Her lægges vagtplanen efter hvilke timetal folk er ansat på. Derudover justerer man antallet af senge til hvor mange sygeplejersker og sosuassistenter der er på arbejde. Ved ferie og sygdom lukkes nogle af sengene, så der på den måde hele tiden er sygeplejersker nok. Dog skal der altid være nogen på arbejde, da hele afdelingen ikke kan lukkes ned. 85 % af de patienter de får ind er akutte patienter, mens godt 15 % er indkaldte.

De timetal folk er ansat på er vidt forskellige og svinger fra 26 til 37 timer. pr. uge. Nogle er også ansat på kommatal, sådan at forstå at man godt kan være ansat til f.eks. 28,3 timer pr. uge. Timetallet bliver overholdt over en 12 ugers periode - dog er det relativt små udsving fra uge til uge, hvilket betyder at timetallene næsten bliver overholdt for hver eneste uge.
Desuden skal der for hver gruppe i afdelingen være mindst en sygeplejerske på arbejde, d.v.s. de må ikke alle være sosu-assistenter.

Det kan ske at folk bliver kaldt ind på tidspunkter hvor de ifølge vagtplanen har fri. Afdelingen har ingen penge til vikarer og kalder derfor istedet folk ind ved sygdom og lign. Det er den ansvarshavende sygeplejerskes ansvar at vurdere om de kan klare sig med det personale der er på arbejde, eller om det er nødvendigt at kalde ekstra personale ind.

A.6.2 Krav til vagplanen

Når vatgplanerne lægges, sørger Lisbet for at overholde overenskomsten, foruden de lokalaftaler der er. Derudover at de prioriterer medarbejdernes ferieønsker meget højt.

I afdelingen er der lavet en lokalaftale, der siger at 11-timersreglen godt må brydes 2 gange pr. 4 uger pr. medarbejder, dog kun med to timer. Det betyder at det godt kan ske, at man kun får 9 timer fri, mellem to på hinanden følgende vagter.

I praksis vil det sige at det kan hænde, at man kan have en aftenvagt efterfulgt af en dagvagt, eller at man har en dagvagt efterfulgt at en nattevagt. Det sker aldrig at at man får en nattevagt efterfulgt af en aftenvagt.

Derudover er det fastlagt at alle skal arbejde hver 3. weekend. På denne måde får alle arbejdet i weekenden, og det passer med at man derved får opfyldt mandskabsbehovet i weekenden.

Desuden er det et krav at der hverdag er en sygeplejerske på arbejde som også var på arbejde dagen før. Dette gøres for at sikre patienterne den bedst mulige pleje, så der er kontinuitet i den information man har om patienterne og hele tiden er en der ved hvad der skete dagen før.

De “onde” vagter fordeles forholdsmæssigt alt efter hvilket timetal man er ansat på.

Der tages ikke hensyn til om folk arbejder godt eller dårligt sammen, hvis de ikke kan sammen, må de som en del af jobbet tage og lære det.

A.6.3 Planlægningsfasen

For at timetallene kan overholdes er der oprettet 53 forskellige vagter. Vagterne er stadig delt ind i dag-, aften- og nattevagt, men der kan godt være mange forskellige af hver slags vagt. For eksempel er der 19 forskellge dagvagter, jvor den tidligste starter klokken 6:30 og den senester klokken 8:45. De har også forskellig varighed (fra 5 til 7.5 time).

Da mandskabsbehovet ikke er fast og afhænger af personalet til rådighed (og tildels af antallet af åbne senge) er der ingen huller i vagtplanen når den lægges.

Når vagtplanen lægges er hele afdelingen delt op i tre grupper. Indenfor hver gruppe sørges der så for at der er en vis mængde personale på arbejde i hver gruppe. Derudover skal der være en ansvarshavende sygeplejerske, når Lisbet er på arbejde er det hende, ellers bliver en af de andre sat til det. Personalet i grupperne roterer hver halve år, så det er ikke faste grupper.

For at holde styr på hvornår hver medarbejder ønsker at holde ferie, eller blot ønsker en enkelt fridag, har Lisbet en "fridags-ønske-bog", hvori medarbejderne skriver deres ferieønsker. Hvis ønskerne er skrevet ind, inden Lisbet lægger vagtplanen for den relevante periode, gør hun meget for at opfylde medarbejdernes ønsker. Dette betyder at nogle til tider kan komme til at arbejde væsentlig mere end de er ansat til, i så fald vil de til gengæld få tilsvarende færre arbejdstimer i den efterfølgende tid.

A.6.4 Fremgangsmåde for vagtplanlægning

De bruger her en 12 ugers rulleplan som skabelon, denne tages så frem hver gang der lægges vagtplan og bliver tilpasset til de uger det nu drejer sig om. Vagtplanerne er altid til rådighed for personalet 4 uger før den tages i brug, i øjeblikket køres der med vagtplaner for 2 uger ad gangen, da dem der var, der galdt for 4 uger ad gangen alligevel aldrig kom til at passe, og altid skulle laves meget om.

Hvis nogen af medarbejderne har haft nogle faste ugentlige ønsker, inden rulleplanen blev lavet, bliver der taget hensyn til dem i rulleplanen. Det kan fx. være at man gerne vil have fri hver tirsdag eller lign.

Hvis en medarbejder har ønske om en fridag, efter Lisbet har lagt vagtplanen for den relevante periode, bliver man nødt til selv at finde en kollega som kan overtage ens vagt. Når man har fundet en kollega, som enten vil bytte en vagt, eller blot overtage en vagt, skal man spørge Lisbet om lov, hvorved hun vurderer om det kan lade sig gøre.

Ud fra rulleplanen, fastlægger Lisbet løbende en-ugers planer, som alle skal være færdige mindst 4 uger inden de tages i brug. Det vil altså sige at rulleplanen i praksis fungerer som skabelon for ugeplanerne, som bliver fastsat afhængig af ønske om fridage osv.

Når man starter med at lægge vagtplanen, startes der altid med de besværlige vagter d.v.s. weekend- og nattevagterne og så aften og dagvagterne.

A.6.5 Ønsker

Lisbet kunne godt tænke sig en vagtplan der var lidt mere fremadrettet, for eksempel at man kunne se hvilke konsekvenser det havde når man gav en sygeplejerske ferie der og der, samt en der øjeblikkelig kunne holde øje med hvor mange feriefridage man har brugt. Det er jo ikke rart at stå i slutningen af april og opdage at halvdelen af personalet mangler at holde to ugers ferie inden det bliver maj. Som det er nu, bliver hun underrettet løbende bagud om folks feriefridage og overtimer, og det ville være rart hele tiden at have til rådighed inde i vagtplanlægningsprogrammet, synes hun. Det er et problem at man skal holde øje med det manuelt.

Ligeledes er proceduren for sygemeldinger lidt tung i det. Og kunne godt gøres lettere. Som det er i dag kan man godt sygemelde en i systemet, men man skal så helst vide hvornår vedkommende er

rask igen, og det kan man jo ikke altid vide.

Bilag B

N-dronningeproblemet

B.1 Constraint Programming

Følgende kode er gengivet fra ECL^{i}PS^e 's hjemmeside (http://www.icparc.ic.ac.uk/eclipse/examples/queens.ecl.txt).

```
% ECLiPSe SAMPLE CODE
%
% AUTHOR:       Joachim Schimpf, IC-Parc
%
% The famous N-queens problem, using finite domains
% and a selection fo different search strategies
%

:- set_flag(gc_interval, 100000000).
:- lib(lists).
:- lib(fd).
:- lib(fd_search).


%--------------------
% The model
%--------------------

queens(N, Board) :-
        length(Board, N),
        Board :: 1..N,
        ( fromto(Board, [Q1|Cols], Cols, []) do
            ( foreach(Q2, Cols), param(Q1), count(Dist,1,_) do
                noattack(Q1, Q2, Dist)
            )
        ).

noattack(Q1,Q2,Dist) :-
        Q2 #\= Q1,
        Q2 - Q1 #\= Dist,
        Q1 - Q2 #\= Dist.


%----------------------
% The search strategies
%----------------------

labeling(a, AllVars) :-
        ( foreach(Var, AllVars) do
            count_backtracks,
            indomain(Var)                       % select value
        ).

labeling(b, AllVars) :-
        ( fromto(AllVars, Vars, VarsRem, []) do
            delete(Var, Vars, VarsRem, 0, first_fail),% dynamic var-select
            count_backtracks,
            indomain(Var)                       % select value
```

```
        ).

labeling(c, AllVars) :-
        middle_first(AllVars, AllVarsPreOrdered), % static var-select
        ( foreach(Var, AllVarsPreOrdered) do
            count_backtracks,
            indomain(Var)                       % select value
        ).

labeling(d, AllVars) :-
        middle_first(AllVars, AllVarsPreOrdered), % static var-select
        ( fromto(AllVarsPreOrdered, Vars, VarsRem, []) do
            delete(Var, Vars, VarsRem, 0, first_fail),% dynamic var-select
            count_backtracks,
            indomain(Var)                       % select value
        ).

labeling(e, AllVars) :-
        middle_first(AllVars, AllVarsPreOrdered), % static var-select
        ( fromto(AllVarsPreOrdered, Vars, VarsRem, []) do
%           search_space(Vars, Size), writeln(Size),
            delete(Var, Vars, VarsRem, 0, first_fail),% dynamic var-select
            count_backtracks,
            indomain(Var, middle)               % select value
        ).


% reorder a list so that the middle elements are first

middle_first(List, Ordered) :-
        halve(List, Front, Back),
        reverse(Front, RevFront),
        splice(Back, RevFront, Ordered).


%----------------------------------
% Toplevel code
%
% all_queens/2 finds all solutions
% first_queens/2 finds one solution
% Strategy is a,b,c,d or e
% N is the size of the board
%----------------------------------

all_queens(Strategy, N) :-              % Find all solutions
        setval(solutions, 0),
        statistics(times, [T0|_]),
        (
            queens(N, Board),
            init_backtracks,
            labeling(Strategy, Board),

%           writeln(Board),
%           put(0'.),
            incval(solutions),
            fail
        ;
            true
        ),
        get_backtracks(B),
        statistics(times, [T1|_]),
        T is T1-T0,
        getval(solutions, S),
        printf("\nFound %d solutions for %d queens in
                  %w s with %d backtracks%n",
                [S,N,T,B]).


first_queens(Strategy, N) :-            % Find one solution
        statistics(times, [T0|_]),
        queens(N, Board),
        statistics(times, [T1|_]),
        D1 is T1-T0,
        printf("Setup for %d queens done in %w seconds", [N,D1]), nl,
        init_backtracks,
        labeling(Strategy, Board),
        get_backtracks(B),
        statistics(times, [T2|_]),
        D2 is T2-T1,
```

```
        printf("Found first solution for %d queens in
130             %w s with %d backtracks%n",
                [N,D2,B]).


135 %--------------------
    % Utilities
    %--------------------

    search_space(Vars, Size) :-
140         ( foreach(V,Vars), fromto(1,S0,S1,Size) do
              dvar_domain(V,D),
              S1 is S0*dom_size(D)
            ).

145
    :- local variable(backtracks), variable(deep_fail).

    init_backtracks :-
            setval(backtracks,0).
150
    get_backtracks(B) :-
            getval(backtracks,B).

    count_backtracks :-
155         setval(deep_fail,false).
    count_backtracks :-
            getval(deep_fail,false),        % may fail
            setval(deep_fail,true),
            incval(backtracks),
160         fail.
```

B.2 Gams

Følgende kode er gengivet fra GAMS' hjemmeside
(http://www.gams.com/modlib/libhtml/queens.htm)

```
 1  $Title Maximum Queens Chess Problem  (QUEENS,SEQ=103)
    $Eolcom !

    $Ontext
 5
       This model finds all possible ways to arrange eight queens on a chess
       board in such a way that no two queens check against any other queen.
       The solution proceeds in two steps. In the first step, we find any
       solution. In the second step we find all solutions by adding cuts to
10     to the original problem set. Finally, a reporting step is added
       to print all solutions found. This problem has a long history. In 1850
       it was investigated by C.F. Gauss, but he was unable to solve it
       completely. There are 92 possible solutions.

15
    Dudeney, H E, Amusements in Mathematics. Dover, New York, 1970.

    Beauvais, J, Solving the Maximum Queens Chess Problem with OSL. IBM
    Kingston, EKKNEWS 2 (1991).
20
    $Offtext

    Sets i  size of chess board     / 1*8 /
         s  diagonal offsets        / 1*13 /   ! 2i-3 diagonals
25  Alias (i,j)

    Parameter sh(s) shift values for diagonals
              rev(i) reverse order;

30  sh(s)  = ord(s) - card(i) + 1 ;
    rev(i) = card(i) + 1 - 2*ord(i);
    Display sh, rev;

    Binary Variable x(i,j) square occupied by Queen
35         Variable tot    total squares occupied by queens

    Equations a(i)  no to queens may be in the same rank
```

```
            b(j)  no to queens may be in the same file
            c(s)  no to queens may be in the same diagonal (forward)
            d(s)  no to queens may be in the same diagonal (backward)
            obj   objective definition ;

    a(i).. sum(j, x(i,j)) =e= 1;

    b(j).. sum(i, x(i,j)) =e= 1;

    c(s).. sum(i, x(i,i+sh(s))) =l= 1;

    d(s).. sum(i, x(i,i+(rev(i)+sh(s)))) =l= 1;

    obj..  tot =e= sum((i,j), x(i,j));

    Model queen1 first model for queens / all /;

    Option optcr = 0;

    Solve queen1 maximizing tot using mip;

    $Stitle Find all remaining solutions

    Sets nn    max number of solutions groups / 1*20 /
         n(nn) dynamic set for solution groups
         t     multiple solutions via rotations and reflections /
               found       original solution
               rot-90      original solution rotated  90 degrees
               rot-180     original solution rotated 180 degrees
               rot-270     original solution rotated 270 degrees
               ref-h       original solution reflected horizontally
               ref-v       original solution reflected vertically
               ref-r       original solution reflected diagonally main
               ref-l       original solution reflected diagonally back   /

    Scalar    saverow, coloff;
    Parameter cutval all possible solutions for cut generation;

    Equation cut(nn,t)  known solutions to be eliminated;

    cut(n,t).. sum((i,j), cutval(n,t,i,j)*x(i,j)) =l= card(i)-1;

    Model queens queens model with cuts / all /;

    Option limrow=0, limcol=0, solprint=off, iterlim = 50000;


    n(nn) = no;  ! clear set of cuts

    queens.solvestat = 1;
    queens.modelstat = 1;

    Loop(nn$(queens.solvestat=1 and queens.modelstat=1),

       n(nn) = yes;   ! add element to set

       cutval(nn,'found'   ,i,j) = x.l(i       ,j       );
       cutval(nn,'rot-90'  ,i,j) = x.l(j+rev(j),i       );
       cutval(nn,'rot-180' ,i,j) = x.l(i+rev(i),j+rev(j));
       cutval(nn,'rot-270' ,i,j) = x.l(j       ,i+rev(i));
       cutval(nn,'ref-h'   ,i,j) = x.l(i+rev(i),j       );
       cutval(nn,'ref-v'   ,i,j) = x.l(i       ,j+rev(j));
       cutval(nn,'ref-r'   ,i,j) = x.l(j       ,i       );
       cutval(nn,'ref-l'   ,i,j) = x.l(j+rev(j),i+rev(i));

       Solve queens maximizing tot using mip;    )
```

Bilag C

Kildekode

C.1 Constraint Programming

C.1.1 `vp.ecl`

```
:- lib(fd_global).
:- lib(matrix_util).
:- lib(clpr).
:- lib(lists).
:- lib(branch_and_bound).
:- lib(repair).

:- local struct(shift(d,e,n,f,s)).
:- local struct(nurseProp(name,night,wh,wishdo)).
:- local struct(cost(shifts, dayoff, subs, weekend, gamssum, sum)).

:- pragma(nodebug).


vp :-

Dem = hil20,
NurseList = hil20,

vp(Dem,NurseList).


vp(help) :-

print("vp(Dem,NurseList,NSubDays,NDiffDays,"),
print("shuffle,Strategy,Timeout,From,To).\n"),
print("vp(Dem,NurseList).\n")
.


vp(Dem,NurseList) :-


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
% Inputdata til modellen:
% 'Dem'     : Behovsmatricen, indikerer hvor stort behovet er
%             for hver vagt for hver dag
% 'NurseList: En matrix indholdende sygeplejserskerne.
%              -et unikt navn
%              -timetal
%              -natkvalifikation
%              -fridagsønsker


%Strategy = continue,
Strategy = step,
%Strategy = dichotomic,
```

```
    Timeout = 1000,
    From = 0,
    To = 2000,

    NSubDays = 28,
    NDiffDays = 2,
    Shuffle=1,

    vp(Dem,NurseList,NSubDays,NDiffDays,
       Shuffle,Strategy,Timeout,From,To).


    vp(Dem,NurseList,NSubDays,NDiffDays,
       Shuffle,Strategy,Timeout,From,To) :-

    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    %
    % Beslutninger vedrørende delløsningerne
    %
    % NSubDays : Antallet af dage en delløsning skal vare
    %            Skal være delelig med 7
    %            Skal kunne gå op i samlede antal dage i perioden
    %
    % NDiffDays: Antallet af dage der ikke overføres fra en delperiode
    %            til den næste delperiode


    % Shuffle:
    %
    %    =1: Hver beslutning i søgningen, sørger for at tage
    %        sygeplejerskerne i en tilfældig rækkefølge
    %    =0: Hver beslutning i søgningen, sørger for at tage
    %        sygeplejerskerne i rækkefølge
    %


    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    %
    % Beslutninger til Branch-and-Bound:
    %
    % Strategy er en af følgende tre:
    %
    %  continue:
    % After finding a solution, continue search
    % with the newly found bound imposed on Cost
    %
    %  step:
    % After finding a solution, restart the whole search
    % with the newly found bound imposed on Cost
    %
    %  dichotomic:
    % After finding a solution, split the remaning cost range
    % and restart search to find a solution in the lower sub-range.
    % If that fails, assume the upper sub-range as the remaning cost
    % range and split again.
    %
    % timeout:
    % number - maximum seconds of cpu time to spend
    %
    % from:
    % number - an initial lower bound for the cost
    %
    % to:
    % number - an initial upper bound for the cost


    cputime(T0),
    ensure_loaded(data),
    ensure_loaded(vp_sub),
    ensure_loaded(vp_search),
    ensure_loaded(vp_misc),

    dim(P,[4]),
    P[shifts of cost]#=1,
    P[dayoff of cost]#=1,
    P[subs of cost]#=1,
    P[weekend of cost]#=1,
```

```
dem(Dem,NNurses,NDays,Demand),
res(NurseList,NNurses,Employees),

% Schedule indeholder vagtplanerne
dim(Schedule,[NNurses,NDays]),

nl,
printf("Antal dage            :               %w\n",NDays),

( for(C,NSubDays,NDays,NSubDays),
  param(Schedule,Demand,Employees,T0,
        Shuffle,Strategy,From,To,Timeout,
        NLabels,NNurses,NDays,NDiffDays,P)
  do
  vp_sub(Schedule,Demand,Employees,OccOfShifts,T0,
         Shuffle,Strategy,From,To,Timeout,
         NLabels,C,NDays,NDiffDays,NNurses,SumCost,Cost,P),

  print_schedule(Schedule,OccOfShifts,Cost),

  CostGList is Cost[1..NNurses,gamssum of cost],
  sumlist(CostGList,SamlPris),
  printf("Samlede pris: %w\n",[SamlPris])
),
  printf("Strategi: %w\n", [Strategy]),

T is cputime-T0,
printf("Benyttet tid: %w\n",[T]),

%delayed_goals(GoalList),
%writeln(GoalList),

nl
.
```

C.1.2 vp_sub.ecl

```
vp_sub(ScheduleOut,Demand,Employees,OccOfShifts,T0,
       Shuffle,Strategy,From,To,Timeout,
       NLabels,NDays,NAllDays,NDiffDays,NNurses,
       SumCost,Cost,P) :-

dim(Schedule,[NNurses,NAllDays]),

( NDays is NAllDays ->
  Schedule = ScheduleOut;
  ( for(NurseC,1,NNurses),
    param(NDays,NDiffDays,Schedule,ScheduleOut)
  do
    ( for(DayC,1,NDays-NDiffDays),
      param(NurseC,Schedule,ScheduleOut)
    do
      Schedule[NurseC,DayC] #= ScheduleOut[NurseC,DayC]
    )
  )
),


Td = 8,      %  07-15
Te = 9.5,    %  15-23
Tn = 9.75,   %  23-07

NLabels=4,

NShifts = NLabels+1,
NWeeks is NDays/7,


make_display_matrix(Schedule, schedule),

dim(OccOfShifts,[NNurses,3]),
dim(Cost,[NNurses,sum of cost]),
dim(MaxShifts,[NNurses,2]),



%Antal dagvagter, aftenvagter, nattevagter
```

```
%og summen af disse i perioden
NDayShifts is sum(Demand[d of shift,1..NDays]),
NEveningShifts is sum(Demand[e of shift,1..NDays]),
NNightShifts is sum(Demand[n of shift,1..NDays]),

%Antal timer til dagvagter, aftenvagter, nattevagtet
%og summen af disse i perioden
NDayDemHours is NDayShifts*Td,
NEveningDemHours is NEveningShifts*Te,
NNightDemHours is NNightShifts*Tn,

NDayEveningDemHours is NDayDemHours + NEveningDemHours,
NAllDemHours is NDayDemHours + NEveningDemHours + NNightDemHours,

%Antal mandetimer til rådighed, fordelt som:
% 1) dem der kan tage nattearbejde og
% 2) alle - timer brugt til nattearbejde
( for(NurseC,1,NNurses),
  fromto(HoursPrNightNurse,[NN|NightNurse], NightNurse, []),
  fromto(HoursPrNurse,[N|Nurse], Nurse, []),
  param(Employees,NWeeks)
do
    NightBool is Employees[night of nurseProp, NurseC],
    Wh_Nurse is Employees[wh of nurseProp, NurseC]*NWeeks,

    ( NightBool is 1 ->
        NN is Wh_Nurse;
        NN is 0
    ),
    N is Wh_Nurse
),

HoursNightNurses is sum(HoursPrNightNurse),
HoursNurses is (sum(HoursPrNurse)-NNightDemHours),
NMoreWork is NDayEveningDemHours-HoursNurses,


printf("Behov (dag+aften)     : %w + %w = %w timer\n",
       [NDayDemHours,NEveningDemHours,NDayEveningDemHours]
      ),
printf("Ressourcer (dag+aften):               %w timer\n",
       [HoursNurses]
      ),
printf("Overarbejde           :               %w timer\n",
       [NMoreWork]
      ),
nl,
printf("Nattearbejde          :               %w timer\n",
       [NNightDemHours]
      ),
nl,


% HARD:                                      %
% En vagttype pr. dag                        %
% Variablerne i Schedule skal være vagttyper %
%                                            %

( for(NurseC,1,NNurses),
  param(NDays,NShifts,Schedule,NLabels)
do
  ( for(DayC,1,1),
    param(NurseC,NLabels,Schedule)
  do
    Schedule[NurseC,DayC] :: 1..NLabels
  ),
  ( for(DayC,2,NDays),
    param(NurseC,NShifts,Schedule)
  do
    Schedule[NurseC,DayC] :: 1..NShifts
  )
),


% HARD:                    %
% Behovet skal være opfyldt %
%                          %

( for(LabelC,1,4),
  param(NDays,NNurses,Demand,Schedule)
```

```
do
  ( for(DayC,1,NDays),
    param(LabelC,NNurses,Demand,Schedule)
  do
    ( LabelC #< 4 ->
      Required is Demand[LabelC,DayC],
      Shifts is Schedule[1..NNurses,DayC],
      occurrences(LabelC,Shifts,Required);

      Required is Demand[f of shift,DayC],
      Shifts is Schedule[1..NNurses,DayC],
      occurrences(f of shift,Shifts,FDays),
      occurrences(s of shift,Shifts,SDays),
      FDays + SDays #= Required
    )
  )
),


% HARD:                                    %
% Find antallet af nattevagter pr ressource %
%                                          %

( for(NurseC,1,NNurses),
  param(Employees,Schedule,OccOfShifts,
        HoursNightNurses,NNightShifts,NDays,NWeeks)
do
  % Max antal nattevagter til denne ressource
  MaxNightShifts_Nurse is
    fix(
      ceiling(
      (
        (Employees[wh of nurseProp,NurseC]*NWeeks)
        /HoursNightNurses
      )                                    %NurseC's brøkdel
      * NNightShifts                       % Antal nattevagter
      * Employees[night of nurseProp,NurseC] % 0 eller 1
      )
    ),
  OccOfShifts[NurseC,n of shift] #= OccN,
  OccN :: [MaxNightShifts_Nurse-1..MaxNightShifts_Nurse],
  OccN #>= 0
),

OccOfNightShifts is OccOfShifts[1..NNurses,n of shift],
sumlist(OccOfNightShifts,NNightShifts),
once labeling(OccOfNightShifts),

print("Ønsket fordeling af vagter:\n"),
print("  Person, Nat,  Dag   ,  Aften \n"),

( for(NurseC,1,NNurses),
  param(Schedule, Employees, OccOfShifts,
        NDays,NAllDays,NNurses, NWeeks,
        Td, Te, Tn,
        Cost,P,
        NDayShifts, NEveningShifts, NNightShifts,
        NDayDemHours, NEveningDemHours,
        NDayEveningDemHours, NAllDemHours,
        HoursNurses,MaxShifts)
do

  printn("  Nr. ",NurseC,":"),

% Den aktuelle ressources vagtplan
% ('shift' er nu den aktuelle række i matricen).
  Shift is Schedule[NurseC,1..NDays],

  occurrences(d of shift,Shift,OccD),
  occurrences(e of shift,Shift,OccE),
  occurrences(n of shift,Shift,OccN),

  OccOfShifts[NurseC,d of shift] #= OccD,
  OccOfShifts[NurseC,e of shift] #= OccE,
  OccOfShifts[NurseC,n of shift] #= OccN,

  printn("  ",OccN,""),
```

```
      % NARD:                     %
      % Mindst 1 fridag per 7 dage %
      %                           %

205     ( for(DayC,1,NDays-6),param(NurseC,Schedule)
        do
          Days is Schedule[NurseC,DayC..DayC+6],
          occurrences(f of shift,Days,DaysOff),
          DaysOff :: 1..7
210     ),

      % HARD:                                                  %
      % 1) Sovedage findes kun efter nattevagter               %
      % 2) Efter nattevagt kommer en nattevagt eller en sovedag %
215   % 3) Dagvagt kan ikke komme efter aftenvagt              %
      %                                                        %

        ( for(DayC,1,NDays-1),
          fromto(Shift, RestVars, RestVars1, _)
220     do
          RestVars = [_|RestVars1],
          RestVars = [SomeDay,NextDay|_],

          SomeDay #= n of shift #/\ NextDay::[n of shift,s of shift] #\/
225       SomeDay #\= n of shift #/\ NextDay #\= s of shift,  %(1

          SomeDay #= e of shift #/\ NextDay #\= d of shift #\/ %(3
          SomeDay #\= e of shift                               %(3
        ),
230

      % HARD:                 %
      % Mindst 2*35 timer fri %
      %  eller 1*55 timer fri %
235   % per hele uge          %
      %                       %

        ( for(DayC,1,NDays-6,7),
          param(Schedule,NurseC,NDays)
240     do

          ( DayC = 1 ->
            FirstDay is DayC;      % DayC -> første mandag i perioden
            FirstDay is DayC-1     % DayC -> foregående uges søndag
245       ),
          ( DayC is NDays-6 ->
            LastDay is DayC+6;     % DayC -> sidste søndag i perioden
            LastDay is DayC+7      % DayC -> efterfølgende uges mandag
          ),
250       DayDiff is LastDay-FirstDay,

          Days is Schedule[NurseC,FirstDay..LastDay],

          ( for(DayC2,1,DayDiff-1),
255         fromto(Days, RestVars, RestVars1, _),
            fromto(HardViolations,[HardViol1|HardViol],HardViol, []),
            param(DayC,DayDiff,NDays,FirstDay)
          do

260         RestVars = [_|RestVars1],
            RestVars = [D1,D2,D3|_],

            % HV1, HV2, HV3: Hviletider, hvor 35<=t<55

265         ( DayC = 1, DayC2 = 1 ->
              % D1,D2,D3 -> Første tre dage i perioden
                      % [F,E,....]
                      % [F,N,....]
              HV1 isd (D1 #= f of shift) #/\
270                   (D2 #= e of shift #\/ D2 #= n of shift);
              HV1 is 0
            ),

                    % [D,F,D,...],[...,D,F,D,...],[...,D,F,D]
275                 % [D,F,E,...],[...,D,F,E,...],[...,D,F,E]
            HV2 isd (D1 #= d of shift                       #/\
                     D2 #= f of shift                       #/\
                    (D3 #= d of shift #\/ D3 #= e of shift)
                    )
280                 #\/
```

```
                % [E,F,E,...],[...,E,F,E,...],[...,E,F,E]
                % [E,F,N,...],[...,E,F,N,...],[...,E,F,N]
                % [S,F,E,...],[...,S,F,E,...],[...,S,F,E]
                % [S,F,N,...],[...,S,F,N,...],[...,S,F,N]
                (
                (D1 #= e of shift #\/ D1 #= s of shift) #/\
                 D2 #= f of shift                      #/\
                (D3 #= e of shift #\/ D3 #= n of shift)
                ),

        ( DayC is NDays-DayDiff+1, DayC2 is DayDiff-1 ->
           % D1,D2,D3 -> sidste tre dage i perioden
                % [......,D,F]
          HV3 isd D2 #= d of shift #/\
                D3 #= f of shift;
          HV3 is 0
        ),


        % HV4: Hviletider, hvor t>=55

        ( FirstDay \= 1, DayC2 = 1 ->
                % [...,D|F,N,...]
          HV4 isd D1 #= d of shift #/\
                  D2 #= f of shift #/\
                  D3 #= n of shift;

          ( DayC is NDays-DayDiff+1, DayC2 is DayDiff-1 ->
                    % D1,D2,D3 -> sidste tre dage i perioden
                    % [...,D,F,N]
            HV4 isd (D1 #= d of shift #/\
                     D2 #= f of shift #/\
                     D3 #= n of shift
                    )
                    #\/
                    % [...,F,F]
                    D2 #= f of shift #/\
                    D3 #= f of shift;

                    % [D,F,N,...],[...,D,F,N,...]
            HV4 isd (D1 #= d of shift #/\
                     D2 #= f of shift #/\
                     D3 #= n of shift
                    )
                    #\/
                    % [F,F,...],[...,F,F,...]
                    D1 #= f of shift #/\
                    D2 #= f of shift
          )
        ),

          HardViol1 #= HV1+HV2+HV3+2*HV4

      ),
    sumlist(HardViolations,SumHV),
    SumHV #>= 2
    ),

  % HARD:              %
  % Hver 2. weekend fri %
  %                    %

    ( for(DayC,1,NDays-13,7),
      param(Schedule,NurseC)
    do
      Days is Schedule[NurseC,DayC..DayC+13],

      Days = [_,_,_,_,_,Sat1,Sun1,_,_,_,_,_,Sat2,Sun2],

      (Sat1 #= f of shift #/\ Sun1 #= f of shift) #\/
      (Sat2 #= f of shift #/\ Sun2 #= f of shift)
    ),

  % SOFT:                     %
  % Vagter skal deles ligeligt %
  %                           %

    Wh_Nurse is Employees[wh of nurseProp, NurseC]*NWeeks,
    ( Wh_Nurse-OccN*Tn >= 0 ->
      NWhDayEvening_Nurse is (Wh_Nurse-OccN*Tn);
```

```
      NWhDayEvening_Nurse is 0
    ),
%  NWhDayEvening_Nurse is (Wh_Nurse-OccN*Tn);

365
    ( for(_,1,1),
      fromto(AViolationsShifts,
             [ViolDMax,ViolEMax,ViolDMin,ViolEMin|ViolationsShifts],
             ViolationsShifts,
370          []
            ),
      param(OccOfShifts,NurseC,
            HoursNurses,NWhDayEvening_Nurse,
            NDayDemHours,NEveningDemHours,
375         Td,Te,Tn,
            NDayShifts,NEveningShifts,MaxShifts)
    do

      MaxDayShifts_Nurse is
380     fix(
          ceiling(
            (
              NWhDayEvening_Nurse/HoursNurses
            ) * NDayShifts
385       )
        ),
      MaxEveningShifts_Nurse is
        fix(
          ceiling(
390         (
              NWhDayEvening_Nurse/HoursNurses
            ) * NEveningShifts
          )
        ),
395
      MaxShifts[NurseC,d of shift] #= MaxDayShifts_Nurse,
      MaxShifts[NurseC,e of shift] #= MaxEveningShifts_Nurse,

      MinDayShifts_Nurse is MaxDayShifts_Nurse-1,
400   MinEveningShifts_Nurse is MaxEveningShifts_Nurse-1,


      ViolD_MaxDiff #= OccOfShifts[NurseC,d of shift]
                       -MaxDayShifts_Nurse,
405   ViolD_MinDiff #= MinDayShifts_Nurse
                       -OccOfShifts[NurseC,d of shift],

      ViolE_MaxDiff #= OccOfShifts[NurseC,e of shift]
                       -MaxEveningShifts_Nurse,
410   ViolE_MinDiff #= MinEveningShifts_Nurse
                       -OccOfShifts[NurseC,e of shift],

      ViolDMax_cost isd OccOfShifts[NurseC,d of shift] #>
                          MaxDayShifts_Nurse,
415   ViolDMin_cost isd OccOfShifts[NurseC,d of shift] #<
                          MinDayShifts_Nurse,

      ViolEMax_cost isd OccOfShifts[NurseC,e of shift] #>
                          MaxEveningShifts_Nurse,
420   ViolEMin_cost isd OccOfShifts[NurseC,e of shift] #<
                          MinEveningShifts_Nurse,


      ViolDMax #= ViolD_MaxDiff*ViolDMax_cost,
425   ViolDMin #= ViolD_MinDiff*ViolDMin_cost,
      ViolEMax #= ViolE_MaxDiff*ViolEMax_cost,
      ViolEMin #= ViolE_MinDiff*ViolEMin_cost,


430   printn(", [",MinDayShifts_Nurse,";"),
      printn("",MaxDayShifts_Nurse,"]"),
      printn(", [",MinEveningShifts_Nurse,";"),
      printn("",MaxEveningShifts_Nurse,"]")

435 ),


  % SOFT:         %
  % Fridagsønsker %
440 %              %
```

```
        HolidayName is Employees[wishdo of nurseProp,NurseC],
        hol(HolidayName,NAllDays,Holidays),

445     (
           for(DayC,1,NDays),
           fromto(Shift, RestVars, RestVars1, _),
           fromto(Holidays,RestDays, RestDays1, _),
           fromto(AViolationsHoliday,[ViolHoliday|ViolationsHoliday],
450               ViolationsHoliday, [])
        do
           RestVars = [_|RestVars1],
           RestDays = [_|RestDays1],

455        RestVars = [SomeDay|_],
           RestDays = [SomeHoliday|_],
           ViolHoliday_tmp isd SomeDay #\= f of shift #/\
                               SomeHoliday #\= 0,

460        ViolHoliday #= ViolHoliday_tmp*SomeHoliday
        ),

     % SOFT:                     %
     % Lørdagsvagt = Søndagsvagt %
465  %                           %

       ( for(DayC,1,NDays,7),
         fromto(Shift, RestVars, RestVars1, _),
         fromto(AViolationsWeekend, [ViolWeekend|ViolationsWeekends],
470             ViolationsWeekends, [])
       do

           RestVars = [_,_,_,_,_,_,_|RestVars1],

475        RestVars = [_,_,_,_,_,Sat,Sun|_],


           ViolWeekend isd (Sat #= d of shift #/\
                            Sun #\= d of shift
480                        ) #\/
                           (Sat #= e of shift #/\
                            Sun #\= e of shift
                           ) #\/
                           (Sat #= n of shift #/\
485                         Sun #\= n of shift
                           ) #\/
                           (
                            (Sat #= f of shift #\/
                             Sat #= s of shift
490                         ) #/\
                            Sun #\= f of shift
                           )
       ),

495
     % SOFT:          %
     % Vagtrækkefølge %
     %                %

500    (
           for(DayC,1,NDays-1),
           fromto(Shift, RestVars, RestVars1, _),
           fromto(AViolationsSubS, [ViolSubS|ViolationsSubS],
505           ViolationsSubS, [])
       do
           RestVars = [_|RestVars1],


           % SOFT: No n after s (s,n)
510        RestVars = [SomeDay,NextDay|_],

           ViolSubS isd SomeDay #= s of shift #/\
                        NextDay #= n of shift
       ),
515

       AVShifts_tmp #= sum(AViolationsShifts),
       Cost[NurseC,shifts of cost]  #=  AVShifts_tmp
     %                                   *AVShifts_tmp
520                                     *P[shifts of cost],
```

```
      AViolationsHoliday_tmp #= sum(AViolationsHoliday),
      Cost[NurseC,dayoff of cost]  #=  AViolationsHoliday_tmp
    %                                  *AViolationsHoliday_tmp
525                                   *P[dayoff of cost],

      AViolationsWeekend_tmp #= sum(AViolationsWeekend),
      Cost[NurseC,weekend of cost] #=  AViolationsWeekend_tmp
    %                                  *AViolationsWeekend_tmp
530                                   *P[weekend of cost],

      AViolationsSubS_tmp #= sum(AViolationsSubS),
      Cost[NurseC,subs of cost]    #=  AViolationsSubS_tmp
    %                                  *AViolationsSubS_tmp
535                                   *P[subs of cost],


      Cost[NurseC,gamssum of cost]  #= Cost[NurseC,shifts of cost]
                                     + Cost[NurseC,dayoff of cost]
540                                  + Cost[NurseC,weekend of cost]
                                     + Cost[NurseC,subs of cost],

      Cost[NurseC,sum of cost]  #= Cost[NurseC,gamssum of cost]
                                  *Cost[NurseC,gamssum of cost],
545   nl
    ),



550 %              %
    % Start søgning %
    %              %
    trimcore,
    Rows is Schedule[1..NNurses,1..NDays],
555 flatten(Rows,Vars),

    CostList is Cost[1..NNurses,sum of cost],
    sumlist(CostList,SumCost),
    maxdomain(SumCost,MaxCost),
560 mindomain(SumCost,MinCost),
    print("Interval for den samlede pris: "),
    printf("[%w;%w]\n",[MinCost,MaxCost]),

    %statistics,
565 %trimcore,
    %statistics,

    make_display_matrix(OccOfShifts, occofshifts),

570 bb_min((nurse_labeling(Schedule,Employees,OccOfShifts,MaxShifts,
                       NNurses,NDays,NAllDays,Shuffle),
            labeling_ffc(Vars)
            %,print_schedule(Schedule,OccOfShifts,Cost)
            ,print_elapsed(T0)
575         ),
            SumCost,
            bb_options with [strategy:Strategy,
                             from:From,
                             to:To,
580                          timeout:Timeout]
          ),


    kill_display_matrix(schedule),
585 kill_display_matrix(occofshifts),


    nl
    .
```

C.1.3 vp_search.ecl

```
1   %---------------------------------------
    % Vores søgestrategi
    %---------------------------------------

5
```

```
nurse_labeling(Schedule,Employees,OccOfShifts,MaxShifts,
               NNurses,NDays,NAllDays,Shuffle) :-


( for(NurseC,1,NNurses),
  foreach(NurseC,NurseList)
do
  true
),

( Shuffle=1 ->
  shuffle(NurseList,NurseListHoliday),
  shuffle(NurseList,NurseListWNS),
  shuffle(NurseList,NurseListWS),
  shuffle(NurseList,NurseListN),
  shuffle(NurseList,NurseListRest);
  NurseListHoliday = NurseList,
  NurseListWNS     = NurseList,
  NurseListWS      = NurseList,
  NurseListN       = NurseList,
  NurseListRest    = NurseList
),


%print("Prøver nu at opfylde fridagsønsker\n"),
% Prøv at opfylde fridagsønsker
( foreach(NurseC,NurseListHoliday),
  param(Employees,Schedule,NDays,NAllDays)
do

  Shift is Schedule[NurseC,1..NDays],
  HolidayName is Employees[wishdo of nurseProp,NurseC],
  hol(HolidayName,NAllDays,Holidays),

  ( for(_,1,NDays),
    fromto(Shift, RestVars, RestVars1, _),
    fromto(Holidays, RestDays, RestDays1, _)
  do

    RestVars = [_|RestVars1],
    RestDays = [_|RestDays1],

    RestVars = [SomeDay|_],
    RestDays = [SomeHoliday|_],

    ( SomeHoliday > 0 ->
      try_day_off(SomeDay);
      true
    )
  )
),
%print_schedule(Schedule),

%print("Prøver nu at bemande weekendnattevagter "),
%print("med samlede vagter\n"),
%Prøv at bemande weekendnattevagter.
( for(DayC,6,NDays-1,7),
  param(Schedule,NurseListWNS,OccOfShifts)
do
  ( foreach(NurseC,NurseListWNS),
    param(Schedule,OccOfShifts,DayC)
  do
    ( OccOfShifts[NurseC,n of shift] >= 2 ->
      try_nightshift(Schedule[NurseC,DayC],
                     Schedule[NurseC,DayC+1]
                    );
      true
    )
  )
),
%print_schedule(Schedule),

%print("Prøver nu at bemande weekender med samlede vagter\n"),
% Prøv at bemande weekender samlet.
( for(DayC,6,NDays-1,7),
  param(Schedule,NurseListWS,OccOfShifts,MaxShifts,NDays)
do
  ( foreach(NurseC,NurseListWS),
```

```
        param(Schedule,OccOfShifts,DayC,MaxShifts,NDays)
      do
        try_eveningshift(Schedule[NurseC,DayC],
                         Schedule[NurseC,DayC+1]
                        ),
          try_dayshift(Schedule[NurseC,DayC],
                       Schedule[NurseC,DayC+1]
                      )
      )
    ),
    %print_schedule(Schedule),

    %print("Prøver nu at bemande resten af nattevagterne\n"),
    % Bemand resterende nattevagter.
    ( for(DayC,1,NDays),
      param(Schedule,NurseListN)
    do
      ( foreach(NurseC,NurseListN),
        param(Schedule,DayC)
      do
        try_nightshift(Schedule[NurseC,DayC])
      )
    ),
    %print_schedule(Schedule),


    %print("Prøver nu at bemande resten af vagterne, dog "),
    %print("således at man maks får 'Max'-1 af hver type\n"),
    ( foreach(NurseC,NurseListRest),
      param(Schedule,MaxShifts,NDays)
    do
      Shifts is Schedule[NurseC,1..NDays],
      flatten(Shifts,FShifts),
      MaxShifts[NurseC,e of shift] #= MSE,
      try_shift_ffc(FShifts,e of shift,MSE),
      MaxShifts[NurseC,d of shift] #= MSD,
      try_shift_ffc(FShifts,d of shift,MSD)
    )
    %,print_schedule(Schedule)
    %,print("Vores søgning er færdig\n")

.
```

C.1.4 `vp_misc.ecl`

```
%---------------
% Helpers


labeling_ff(AllVars) :-
( fromto(AllVars, Vars, Rest, [])
do
  deleteff(X, Vars, Rest),
  indomain(X)
).

labeling_ffc(AllVars) :-
( fromto(AllVars, Vars, Rest, [])
do
  deleteffc(X, Vars, Rest),
  indomain(X)
).

try_shift_ffc(AllVars,SType,MS) :-
( fromto(AllVars, Vars, Rest, []),
  param(AllVars,SType,MS)
do
  occ_shift(AllVars,SType,N),
  ( N < MS-1 ->
    deleteffc(X, Vars, Rest),
    try_shift(X,SType);
    true
  )
).

occ_shift(Shifts,SType,N) :-
```

```
  length(Shifts, NDays),
  ( for(DayC,1,NDays),
    fromto(Shifts, RestVars, RestVars1, _),
    fromto(ANDayShifts, [NDS|NDayShifts],NDayShifts, []),
    param(SType)
  do
    RestVars = [_|RestVars1],
    RestVars = [SomeDay|_],

    dvar_domain(SomeDay,Dom),
    dom_range(Dom,DMin,DMax),

    NDS isd DMin #= DMax #/\ DMax #= SType
  ),
  sumlist(ANDayShifts,N)
.


print_elapsed(T0) :-
 T is cputime-T0,
 printf("Benyttet tid: %w\n",[T])
.

print_costs(SumCost,NNurses,Cost) :-
    printf("Samlet pris:                    %w \n",
           [SumCost]),
    CostList is Cost[1..NNurses, sum of cost],
    printf("Samlet pris pr. medarbejder:    %w \n\n",
            [CostList]),

    CostListShifts is Cost[1..NNurses, shifts of cost],
    printf(" -pris for antallet af vagter:  %w \n",
            [CostListShifts]),
    CostListWeekend is Cost[1..NNurses, weekend of cost],
    printf(" -pris for weekendsammenlægning: %w \n",
            [CostListWeekend]),
    CostListHol is Cost[1..NNurses, dayoff of cost],
    printf(" -pris for ferieønsker:         %w \n",
            [CostListHol]),
    CostListSubs is Cost[1..NNurses, subs of cost],
    printf(" -pris for efterfølgende vagter: %w \n",
            [CostListSubs])
.


print_schedule(Schedule) :-

  dim(Schedule, [NNurses,NDays]),

  print("       "),
  ( for(W,1,NDays//7) do
    print(" m  t  o  t  f  l  s ")
  ),
  nl,

  printl(NDays),

  nl,

  ( for(NurseC,1,NNurses), param(Schedule,NDays)
  do
    printn("Nr.",NurseC,": "),
    ( for(Day,1,NDays), param(NurseC,Schedule)
    do
      Table = shift with [f:'F',d:'D',e:'A',n:'N',s:'S'],
      Post is Schedule[NurseC,Day],
      ( var(Post) -> Symbol= - ;
      Symbol is Table[Schedule[NurseC,Day]]),
      printf(" %w ",[Symbol])
    ),
    print("| "),
    mod(NurseC,5,X),
    ( X=0 -> nl, printl(NDays) ; true ),
    nl
  ),
  printl(NDays),
  nl
.
```

```
print_schedule(Schedule,OccOfShifts,Cost) :-

  dim(Schedule, [NNurses,NDays]),
  dim(OccOfShifts, [NNurses,NLabels]),

  print("      "),
  ( for(W,1,NDays//7) do
    print(" m  t  o  t  f  l  s ")
  ),
  print("|  D  A  N | AV WS FØ EV  sum |"),
  nl,

  printl(NDays),

  nl,

  ( for(NurseC,1,NNurses),
    param(Schedule,OccOfShifts,Cost,NDays,NLabels)
  do
    printn("Nr.",NurseC,": "),
    ( for(Day,1,NDays), param(NurseC,Schedule)
    do
      Table = shift with [f:'F',d:'D',e:'A',n:'N',s:'S'],
      Post is Schedule[NurseC,Day],
      ( var(Post) -> Symbol= - ;
        Symbol is Table[Schedule[NurseC,Day]]),
      printf(" %w ",[Symbol])
    ),
    print("| "),
    ( for(Label,1,NLabels), param(OccOfShifts,NurseC)
     do
      Shift is OccOfShifts[NurseC,Label],
      printn("",Shift," ")
    ),
    print("| "),
    AV is Cost[NurseC, shifts of cost],
    printn("",AV," "),
    WS is Cost[NurseC, weekend of cost],
    printn("",WS," "),
    FO is Cost[NurseC, dayoff of cost],
    printn("",FO," "),
    EV is Cost[NurseC, subs of cost],
    printn("",EV," "),
    SUM is Cost[NurseC, gamssum of cost],
    printn(" ",SUM," "),
    print(" | "),
    mod(NurseC,5,X),
    ( X=0 -> nl, printl(NDays) ; true ),
    nl
  ),
  printl(NDays),
  nl,

  print("AV: 'Antal vagter'\n"),
  print("WS: 'Weekendsammenlægning'\n"),
  print("FØ: 'Fridagsønsker'\n"),
  print("EV: 'Efterfølgende vagter'\n"),
  nl
.

print_occofshifts(OccOfShifts) :-
        dim(OccOfShifts, [NNurses,NLabels]),
        print("\nAntal vagter:\n"),
        print(" D  A  N F/S\n"),
        print(" -----------\n"),
        ( for(Res,1,NNurses), param(OccOfShifts,NNurses,NLabels) do
          ( for(Label,1,NLabels), param(OccOfShifts,Res) do
              Shift is OccOfShifts[Res,Label],
              printf(" %w ",[Shift])
          ),
          nl
        ),
        nl
.

printn(H,Number,T) :-
  ( Number < 10 ->
      printf("%w %w%w",[H,Number,T]);
      printf("%w%w%w",[H,Number,T])
  ).
```

```
printl(NDays) :-
  print("-------"),
  ( for(_,1,NDays//7)
  do
    print("---------------------")
  ),
  print("|----------|------------------|")
.

try_shift(X,SType) :- X #= SType.
try_shift(_,_) :- true.

try_nightshift(X) :-  X #= n of shift.
try_nightshift(X) :-  X #\= n of shift.

%try_nightshift(X,Y) :-  X #= n of shift #/\ Y #= n of shift.
try_nightshift(X,Y) :-  X #= Y #/\ Y #= n of shift.
try_nightshift(_,_) :- true.

try_dayshift(X) :- X #= d of shift.
try_dayshift(_) :- true. % :-  X #\= d of shift.

try_dayshift(X,Y) :-  X #= d of shift #/\ Y #= d of shift.
try_dayshift(_,_) :-  true.

try_eveningshift(X) :- X #= e of shift.
try_eveningshift(_) :- true. % :-  X #\= e of shift.

try_eveningshift(X,Y) :-  X #= e of shift #/\ Y #= e of shift.
try_eveningshift(_,_) :-  true.

try_day_off(X) :- X #= f of shift.
try_day_off(_) :- true.
```

C.1.5 `data.ecl`

```
%---------------
% Data

%                %
% Behovsmatricer %
%                %
%
% Behovsmatricerne læses således:
%
% dem('navn', 'antal medarbejdere', 'antal dage', shift with [
%    f: demand ( ... )
%    d: demand ( ... )   <- lister indholdende behov per
%    e: demand ( ... )      dag, for hhv. fri+sovedag,
%    n: demand ( ... )      dagvagt, aftenvagt og nattevagt
% ])



% Inspireret fra Hillerød Sygehus
dem(hil18, 18, 28, shift with [
   f: demand( 9, 9, 9, 9, 9,10,10, 9, 9, 9, 9, 9,10,10,
              9, 9, 9, 9, 9,10,10, 9, 9, 9, 9, 9,10,10),
   d: demand( 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3,
              3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3),
   e: demand( 4, 4, 4, 4, 4, 3, 3, 4, 4, 4, 4, 4, 3, 3,
              4, 4, 4, 4, 4, 3, 3, 4, 4, 4, 4, 4, 3, 3),
   n: demand( 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2,
              2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2)
]).

% Inspireret fra Hillerød Sygehus
dem(hil20, 20, 28, shift with [
   f: demand(11,11,11,11,11,12,12,11,11,11,11,11,12,12,
             11,11,11,11,11,12,12,11,11,11,11,11,12,12),
   d: demand( 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3,
              3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3),
   e: demand( 4, 4, 4, 4, 4, 3, 3, 4, 4, 4, 4, 4, 3, 3,
              4, 4, 4, 4, 4, 3, 3, 4, 4, 4, 4, 4, 3, 3),
   n: demand( 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2,
              2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2)
```

```
]).
% Inspireret fra skadestuen på Holbæk sygehus
dem(hol13,13,28, shift with [
   f: demand( 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8,
              8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8),
   d: demand( 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2,
              2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2),
   e: demand( 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2,
              2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2),
   n: demand( 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
              1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1)
]).

dem(var13b,13,28, shift with [
   f: demand( 8, 7, 6, 5, 8, 8,13, 6, 8,10, 5, 8, 9,11,
              7,10, 8, 7, 9, 8, 8, 8,10, 6, 8, 7, 9, 9),
   d: demand( 2, 3, 3, 3, 0, 1, 0, 4, 5, 1, 2, 2, 2, 0,
              2, 1, 2, 2, 1, 3, 2, 2, 2, 4, 2, 4, 1, 2),
   e: demand( 2, 2, 3, 3, 3, 1, 0, 2, 0, 1, 4, 3, 2, 2,
              2, 1, 2, 2, 3, 2, 2, 2, 0, 2, 2, 0, 2, 1),
   n: demand( 1, 1, 1, 2, 2, 3, 0, 1, 0, 1, 1, 1, 0, 0, 0,
              2, 1, 1, 2, 1, 0, 1, 0, 1, 1, 1, 0, 1, 1)
]).

dem(hol13u8,13,56, shift with [
   f: demand( 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8,
              8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8,
              8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8,
              8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8),
   d: demand( 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2,
              2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2,
              2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2,
              2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2),
   e: demand( 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2,
              2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2,
              2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2,
              2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2),
   n: demand( 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
              1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
              1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
              1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1)
]).


dem(hol130,130,28, shift with [
   f: demand(80,80,80,80,80,80,80,80,80,80,80,80,80,80,
             80,80,80,80,80,80,80,80,80,80,80,80,80,80),
   d: demand(20,20,20,20,20,20,20,20,20,20,20,20,20,20,
             20,20,20,20,20,20,20,20,20,20,20,20,20,20),
   e: demand(20,20,20,20,20,20,20,20,20,20,20,20,20,20,
             20,20,20,20,20,20,20,20,20,20,20,20,20,20),
   n: demand(10,10,10,10,10,10,10,10,10,10,10,10,10,10,
             10,10,10,10,10,10,10,10,10,10,10,10,10,10)
]).

dem(sim10,10,28, shift with [
   f: demand( 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7,
              7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7),
   d: demand( 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
              1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1),
   e: demand( 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
              1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1),
   n: demand( 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
              1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1)
]).


%                        %
% Medarbejderoplysninger %
%                        %
%
% Ressourcematricerne læses således:
%
% res('navn', 'antal medarbejdere', nurseProp with [
%     name:  employees( ... ) <- Medarbejdernes navne
%     night: employees( ... ) <- Natkvalifikation (1 eller 0)
%     wh:    employees( ... ) <- Timetal
%     wisdo: employees( ... ) <- Navn på fridagsønskeliste
% ])
```

```
% Inspireret fra Hillerød Sygehus
res(hil18,18, nurseProp with [
    name:    employees(  n1,  n2,  n3,  n4,  n5,  n6,  n7,  n8,
                         n9, n10, n11, n12, n13, n14, n15, n16,
                        n17, n18),
    night:   employees(   1,   1,   1,   1,   0,   1,   1,   1,
                          1,   1,   1,   1,   1,   1,   1,   1,
                          1,   1),
    wh:      employees(31.7,31.1,30.9,27.6,27.6,18.05,27.8,23.9,
                       23.9,23.9,27.6,27.4,27.2,24.05,18.2,26.9,
                       30.1,27),
    wishdo:  employees(  no,  no,  no,  no,  no,  no,  no,  no,
                         no,  no,  no,  no,  no,  no,  no,  no,
                         no,  no)
]).

% Inspireret fra Hillerød Sygehus
res(hil20,20, nurseProp with [
    name:    employees(  n1,  n2,  n3,  n4,  n5,  n6,  n7,  n8,
                         n9, n10, n11, n12, n13, n14, n15, n16,
                        n17, n18, n19, n20),
    night:   employees(   1,   1,   1,   1,   0,   1,   1,   1,
                          1,   1,   1,   1,   1,   1,   1,   1,
                          1,   1,   1,   1),
    wh:      employees(31.7,31.1,30.9,27.6,27.6,18.05,27.8,23.9,
                       23.9,23.9,27.6,27.4,27.2,24.05,18.2,26.9,
                       30.1,27  , 27 , 27 ),
    wishdo:  employees(  no,  no,  no,  no,  no,  no,  no,  no,
                         no,  no,  no,  no, no, no, no, no, no,
                         no,  no,  no,  no)
        ]).

% Inspireret fra skadestuen på Holbæk sygehus
res(hol13,13, nurseProp with [
    name:    employees(  n1,  n2,  n3,  n4,  n5,  n6,  n7,  n8,
                         n9, n10, n11, n12, n13),
    night:   employees(   1,   1,   1,   1,   1,   1,   1,   1,
                          1,   1,   1,   1,   1),
    wh:      employees(  20,  20,  20,  20,  20,  20,  20,  20,
                         20,  20,  20,  20,  20),
    wishdo:  employees(  no,  no,  no,  no,  no,  no,  no,  no,
                         no,  no,  no,  no,  no)
        ]).

res(hol13t2,13, nurseProp with [
    name:    employees(  n1,  n2,  n3,  n4,  n5,  n6,  n7,  n8,
                         n9, n10, n11, n12, n13),
    night:   employees(   1,   1,   1,   1,   1,   1,   1,   1,
                          1,   1,   1,   1,   1),
    wh:      employees(  10,  10,  10,  10,  10,  10,  10,  20,
                         20,  20,  20,  20,  20),
    wishdo:  employees(  no,  no,  no,  no,  no,  no,  no,  no,
                         no,  no,  no,  no,  no)
        ]).


res(hol13n2,13, nurseProp with [
    name:    employees(  n1,  n2,  n3,  n4,  n5,  n6,  n7,  n8,
                         n9, n10, n11, n12, n13),
    night:   employees(   0,   0,   0,   0,   0,   0,   0,   1,
                          1,   1,   1,   1,   1),
    wh:      employees(  20,  20,  20,  20,  20,  20,  20,  20,
                         20,  20,  20,  20,  20),
    wishdo:  employees(  no,  no,  no,  no,  no,  no,  no,  no,
                         no,  no,  no,  no,  no)
        ]).


res(hol13f,13, nurseProp with [
    name:    employees(  n1,  n2,  n3,  n4,  n5,  n6,  n7,  n8,
                         n9, n10, n11, n12, n13),
    night:   employees(   1,   1,   1,   1,   1,   1,   1,   1,
                          1,   1,   1,   1,   1),
    wh:      employees(  20,  20,  20,  20,  20,  20,  20,  20,
                         20,  20,  20,  20,  20),
    wishdo:  employees(  no,  no,  n3,  n4,  no,  no,  n7,  no,
                         no,  no, n11,  no,  no)
        ]).
```

```
res(hol13f1,13, nurseProp with [
    name:    employees(  n1,  n2,  n3,  n4,  n5,  n6,  n7,  n8,
                         n9, n10, n11, n12, n13),
    night:   employees(   1,   1,   1,   1,   1,   1,   1,   1,
                          1,   1,   1,   1,   1),
    wh:      employees(  20,  20,  20,  20,  20,  20,  20,  20,
                         20,  20,  20,  20,  20),
    wishdo:  employees(  t2,  t2,  t2,  t2,  t2,  t2,  t2,  t2,
                         t2,  t2,  t2,  t2,  t2)
        ]).


res(hol13fa,13, nurseProp with [
    name:    employees(  n1,  n2,  n3,  n4,  n5,  n6,  n7,  n8,
                         n9, n10, n11, n12, n13),
    night:   employees(   1,   1,   1,   1,   1,   1,   1,   1,
                          1,   1,   1,   1,   1),
    wh:      employees(  20,  20,  20,  20,  20,  20,  20,  20,
                         20,  20,  20,  20,  20),
    wishdo:  employees(  n1,  n2,  n3,  n4,  n5,  n6,  n7,  n8,
                         n9, n10, n11, n12, n13)
        ]).

res(hol130,130, nurseProp with [
    name:    employees(  n1,  n2,  n3,  n4,  n5,  n6,  n7,  n8,
                         n9, n10, n11, n12, n13, n14, n15, n16,
                        n17, n18, n19, n20, n21, n22, n23, n24,
                        n25, n26, n27, n28, n29, n30, n31, n32,
                        n33, n34, n35, n36, n37, n38, n39, n40,
                        n41, n42, n43, n44, n45, n46, n47, n48,
                        n49, n50, n51, n52, n53, n54, n55, n56,
                        n57, n58, n59, n60, n61, n62, n63, n64,
                        n65, n66, n67, n68, n69, n70, n71, n72,
                        n73, n74, n75, n76, n77, n78, n79, n80,
                        n81, n82, n83, n88, n85, n86, n87, n88,
                        n89, n90, n91, n92, n93, n94, n95, n96,
                        n97, n98, n99,n100,n101,n102,n103,n104,
                       n105,n106,n107,n108,n109,n100,n111,n112,
                       n113,n114,n115,n116,n117,n118,n119,n120,
                       n121,n122,n123,n124,n125,n126,n127,n128,
                       n129,n130),
    night:   employees(   1,   1,   1,   1,   1,   1,   1,   1,
                          1,   1,   1,   1,   1,   1,   1,   1,
                          1,   1,   1,   1,   1,   1,   1,   1,
                          1,   1,   1,   1,   1,   1,   1,   1,
                          1,   1,   1,   1,   1,   1,   1,   1,
                          1,   1,   1,   1,   1,   1,   1,   1,
                          1,   1,   1,   1,   1,   1,   1,   1,
                          1,   1,   1,   1,   1,   1,   1,   1,
                          1,   1,   1,   1,   1,   1,   1,   1,
                          1,   1,   1,   1,   1,   1,   1,   1,
                          1,   1,   1,   1,   1,   1,   1,   1,
                          1,   1,   1,   1,   1,   1,   1,   1,
                          1,   1,   1,   1,   1,   1,   1,   1,
                          1,   1,   1,   1,   1,   1,   1,   1,
                          1,   1,   1,   1,   1,   1,   1,   1,
                          1,   1,   1,   1,   1,   1,   1,   1,
                          1,   1),
    wh:      employees(  20,  20,  20,  20,  20,  20,  20,  20,
                         20,  20,  20,  20,  20,  20,  20,  20,
                         20,  20,  20,  20,  20,  20,  20,  20,
                         20,  20,  20,  20,  20,  20,  20,  20,
                         20,  20,  20,  20,  20,  20,  20,  20,
                         20,  20,  20,  20,  20,  20,  20,  20,
                         20,  20,  20,  20,  20,  20,  20,  20,
                         20,  20,  20,  20,  20,  20,  20,  20,
                         20,  20,  20,  20,  20,  20,  20,  20,
                         20,  20,  20,  20,  20,  20,  20,  20,
                         20,  20,  20,  20,  20,  20,  20,  20,
                         20,  20,  20,  20,  20,  20,  20,  20,
                         20,  20,  20,  20,  20,  20,  20,  20,
                         20,  20,  20,  20,  20,  20,  20,  20,
                         20,  20,  20,  20,  20,  20,  20,  20,
                         20,  20,  20,  20,  20,  20,  20,  20,
                         20,  20),
    wishdo:  employees(  no,  no,  no,  no,  no,  no,  no,  no,
                         no,  no,  no,  no,  no,  no,  no,  no,
                         no,  no,  no,  no,  no,  no,  no,  no,
                         no,  no,  no,  no,  no,  no,  no,  no,
```

```
                            no,  no,  no,  no,  no,  no,  no,  no,
                            no,  no,  no,  no,  no,  no,  no,  no,
                            no,  no,  no,  no,  no,  no,  no,  no,
                            no,  no,  no,  no,  no,  no,  no,  no,
                            no,  no,  no,  no,  no,  no,  no,  no,
                            no,  no,  no,  no,  no,  no,  no,  no,
                            no,  no,  no,  no,  no,  no,  no,  no,
                            no,  no,  no,  no,  no,  no,  no,  no,
                            no,  no,  no,  no,  no,  no,  no,  no,
                            no,  no,  no,  no,  no,  no,  no,  no,
                            no,  no)
        ]).


res(sim10,10, nurseProp with [
    name:    employees(  n1,  n2,  n3,  n4,  n5,  n6,  n7,  n8,
                         n9, n10),
    night:   employees(   1,   1,   1,   1,   1,   1,   1,   1,
                          1,   1),
    wh:      employees(  20,  20,  20,  20,  20,  20,  20,  20,
                         20,  20),
    wishdo:  employees(  no,  no,  no,  no,  no,  no,  no,  no,
                         no,  no)
        ]).


%             %
% Ferieønsker %
%             %
%
% Ferieønskelisterne læses således:
%
% hol('navn', 'antal dag', [ ... ]).
%    indholdet i den sidste liste, er et tal per dag,
%    hvor en værdi på mere end 0, angiver at man ønsker
%    fri. Jo større værdi,  jo mere ønskes der fri

%             m,t,o,t,f,l,s,m,t,o,t,f,l,s
hol(no,28,   [0,0,0,0,0,0,0,0,0,0,0,0,0,0,
              0,0,0,0,0,0,0,0,0,0,0,0,0,0]).

hol(n1,28,   [0,0,0,0,0,0,0,0,0,0,0,0,0,0,
              0,0,0,0,0,0,0,0,0,0,0,0,0,0]).

hol(n2,28,   [0,2,0,0,0,1,0,0,0,0,0,0,0,0,
              0,0,0,0,0,0,0,2,0,0,0,0,0,0]).

hol(n3,28,   [0,0,0,0,0,1,0,0,0,0,1,0,0,0,
              0,0,0,2,0,0,0,0,0,0,2,0,0,0]).

hol(n4,28,   [0,0,0,2,2,2,0,0,0,0,0,0,0,0,
              0,0,0,0,0,0,0,0,2,2,2,0,0,0]).

hol(n5,28,   [0,1,0,0,0,0,0,0,0,0,0,1,0,0,
              0,0,0,0,0,0,0,0,0,0,0,0,0,1]).

hol(n6,28,   [1,0,0,0,1,0,0,0,0,0,0,0,0,0,
              0,1,0,0,0,1,0,0,0,0,0,0,0,0]).

hol(n7,28,   [0,0,0,0,0,0,0,0,0,0,0,0,0,0,
              0,0,0,0,2,2,1,2,0,0,0,0,0,0]).

hol(n8,28,   [0,0,1,2,0,0,0,0,0,0,1,0,0,0,
              0,2,0,0,2,0,0,0,0,0,0,0,0,0]).

hol(n9,28,   [0,0,0,0,0,0,0,0,0,0,0,0,0,0,
              0,0,0,0,0,0,0,0,0,0,0,0,0,0]).

hol(n10,28,  [0,0,0,1,1,1,1,1,0,0,0,0,0,0,
              0,0,0,0,0,0,0,0,0,0,0,0,0,0]).

hol(n11,28,  [0,0,2,0,0,0,0,0,0,2,0,0,0,0,
              0,0,2,0,0,0,0,0,0,2,0,0,0,0]).
```

```
    hol(n12,28,  [0,0,0,0,0,0,0,0,0,0,0,0,0,0,
                  0,0,0,0,0,0,0,0,0,0,0,0,0,0]).

    hol(n13,28,  [0,0,0,0,0,0,0,0,0,0,0,0,0,0,
                  0,0,0,0,0,0,0,0,0,1,1,1,1,1]).

    hol(t2,28,   [0,0,0,0,0,0,0,0,1,0,0,0,0,0,
                  0,0,0,0,0,0,0,0,0,0,0,0,0,0]).

    hol(no,56,   [0,0,0,0,0,0,0,0,0,0,0,0,0,0,
                  0,0,0,0,0,0,0,0,0,0,0,0,0,0,
                  0,0,0,0,0,0,0,0,0,0,0,0,0,0,
                  0,0,0,0,0,0,0,0,0,0,0,0,0,0]).
```

C.1.6 Output

Følgende er et eksempel på en programkørsel af CP-programmet.

```
[eclipse 43]: vp(hol13, hol13, 14, 2, 0, step, 500, 0, 2000).

Antal dage           :                28
Behov (dag+aften)    : 224 + 266.0 = 490.0 timer
Ressourcer (dag+aften):               383.5 timer
Overarbejde          :                106.5 timer

Nattearbejde         :                136.5 timer

Ønsket fordeling af vagter:
  Person, Nat,  Dag   ,  Aften
  Nr.  1:   1, [ 2; 3], [ 2; 3]
  Nr.  2:   1, [ 2; 3], [ 2; 3]
  Nr.  3:   1, [ 2; 3], [ 2; 3]
  Nr.  4:   1, [ 2; 3], [ 2; 3]
  Nr.  5:   1, [ 2; 3], [ 2; 3]
  Nr.  6:   1, [ 2; 3], [ 2; 3]
  Nr.  7:   1, [ 2; 3], [ 2; 3]
  Nr.  8:   1, [ 2; 3], [ 2; 3]
  Nr.  9:   1, [ 2; 3], [ 2; 3]
  Nr. 10:   1, [ 2; 3], [ 2; 3]
  Nr. 11:   1, [ 2; 3], [ 2; 3]
  Nr. 12:   1, [ 2; 3], [ 2; 3]
  Nr. 13:   2, [ 1; 2], [ 1; 2]
Interval for den samlede pris: [0;20800]
Benyttet tid: 1.0
Found a solution with cost 2
Branch-and-bound timeout!

       m  t  o  t  f  l  s  m  t  o  t  f  l  s  m  t  o  t  f  l  s  m  t  o  t  f  l  s |  D  A  N | AV WS FØ EV  sum |
-------------------------------------------------------------------------------------------------------------------------
Nr. 1:  N  S  F  F  D  A  A  F  D  F  F  F  -  -  -  -  -  -  -  -  -  -  -  -  -  -  -  -  -  -  -  -  -  -  -  -  -  |  2  2  1 |  0  0  0  0   0  |
Nr. 2:  F  N  S  F  F  A  A  F  D  D  F  D  -  -  -  -  -  -  -  -  -  -  -  -  -  -  -  -  -  -  -  -  -  -  -  -  -  |  3  2  1 |  0  0  0  0   0  |
Nr. 3:  F  F  N  S  F  D  D  F  A  A  F  F  -  -  -  -  -  -  -  -  -  -  -  -  -  -  -  -  -  -  -  -  -  -  -  -  -  |  2  2  1 |  0  0  0  0   0  |
Nr. 4:  F  A  F  N  S  D  D  F  A  A  F  F  -  -  -  -  -  -  -  -  -  -  -  -  -  -  -  -  -  -  -  -  -  -  -  -  -  |  2  3  1 |  0  0  0  0   0  |
Nr. 5:  A  F  F  F  D  F  F  N  S  D  F  F  -  -  -  -  -  -  -  -  -  -  -  -  -  -  -  -  -  -  -  -  -  -  -  -  -  |  2  3  1 |  0  0  0  0   0  |
-------------------------------------------------------------------------------------------------------------------------
Nr. 6:  A  F  F  D  F  F  F  D  N  S  F  F  -  -  -  -  -  -  -  -  -  -  -  -  -  -  -  -  -  -  -  -  -  -  -  -  -  |  2  3  1 |  0  0  0  0   0  |
Nr. 7:  F  F  F  A  A  F  F  F  F  N  S  F  -  -  -  -  -  -  -  -  -  -  -  -  -  -  -  -  -  -  -  -  -  -  -  -  -  |  2  2  1 |  0  0  0  0   0  |
Nr. 8:  F  F  F  F  A  F  F  A  F  F  N  S  -  -  -  -  -  -  -  -  -  -  -  -  -  -  -  -  -  -  -  -  -  -  -  -  -  |  2  2  1 |  0  0  0  0   0  |
Nr. 9:  F  F  D  A  N  S  F  A  F  F  D  F  -  -  -  -  -  -  -  -  -  -  -  -  -  -  -  -  -  -  -  -  -  -  -  -  -  |  2  2  1 |  0  0  0  0   0  |
Nr.10:  D  D  A  F  F  F  F  D  F  F  A  N  -  -  -  -  -  -  -  -  -  -  -  -  -  -  -  -  -  -  -  -  -  -  -  -  -  |  3  2  1 |  0  0  0  0   0  |
-------------------------------------------------------------------------------------------------------------------------
Nr.11:  F  F  D  D  F  F  F  F  F  F  A  A  -  -  -  -  -  -  -  -  -  -  -  -  -  -  -  -  -  -  -  -  -  -  -  -  -  |  2  2  1 |  0  1  0  0   1  |
Nr.12:  D  D  A  F  F  F  F  F  F  F  D  A  -  -  -  -  -  -  -  -  -  -  -  -  -  -  -  -  -  -  -  -  -  -  -  -  -  |  3  2  1 |  0  1  0  0   1  |
Nr.13:  F  A  F  F  F  N  N  S  F  F  F  D  -  -  -  -  -  -  -  -  -  -  -  -  -  -  -  -  -  -  -  -  -  -  -  -  -  |  1  1  2 |  0  0  0  0   0  |
-------------------------------------------------------------------------------------------------------------------------
AV: 'Antal vagter'
WS: 'Weekendsammenlægning'
FØ: 'Fridagsønsker'
EV: 'Efterfølgende vagter'

Samlede pris: 2
Behov (dag+aften)    : 448 + 532.0 = 980.0 timer
Ressourcer (dag+aften):               767.0 timer
Overarbejde          :                213.0 timer

Nattearbejde         :                273.0 timer

Ønsket fordeling af vagter:
  Person, Nat,  Dag   ,  Aften
  Nr.  1:   2, [ 4; 5], [ 4; 5]
  Nr.  2:   2, [ 4; 5], [ 4; 5]
  Nr.  3:   2, [ 4; 5], [ 4; 5]
  Nr.  4:   2, [ 4; 5], [ 4; 5]
  Nr.  5:   2, [ 4; 5], [ 4; 5]
  Nr.  6:   2, [ 4; 5], [ 4; 5]
  Nr.  7:   2, [ 4; 5], [ 4; 5]
  Nr.  8:   2, [ 4; 5], [ 4; 5]
  Nr.  9:   2, [ 4; 5], [ 4; 5]
  Nr. 10:   2, [ 4; 5], [ 4; 5]
  Nr. 11:   2, [ 4; 5], [ 4; 5]
  Nr. 12:   3, [ 3; 4], [ 3; 4]
  Nr. 13:   3, [ 3; 4], [ 3; 4]
Interval for den samlede pris: [0;24059]
Benyttet tid: 503.25
Found a solution with cost 13
Benyttet tid: 505.1
Found a solution with cost 10
Benyttet tid: 506.940000000001
Found a solution with cost 9
Benyttet tid: 508.810000000001
Found a solution with cost 8
Benyttet tid: 510.66
Found a solution with cost 7
Benyttet tid: 512.490000000002
Found a solution with cost 6
Benyttet tid: 515.470000000001
Found a solution with cost 5
Benyttet tid: 561.210000000001
Found a solution with cost 4
Benyttet tid: 563.08
Found a solution with cost 3
Benyttet tid: 564.990000000002
Found a solution with cost 2
Branch-and-bound timeout!

       m  t  o  t  f  l  s  m  t  o  t  f  l  s  m  t  o  t  f  l  s  m  t  o  t  f  l  s |  D  A  N | AV WS FØ EV  sum |
-------------------------------------------------------------------------------------------------------------------------
Nr. 1:  N  S  F  F  D  A  A  F  D  F  F  F  F  N  S  F  F  D  A  A  F  F  D  F  F  F  F  |  4  4  2 |  0  0  0  0   0  |
Nr. 2:  F  N  S  F  F  A  A  F  D  D  F  D  F  F  A  N  S  F  F  A  A  F  D  F  F  D  F  F  |  5  5  2 |  0  0  0  0   0  |
Nr. 3:  F  F  N  S  F  D  D  F  A  A  F  F  F  F  F  N  S  F  D  D  A  A  F  F  D  F  F  |  5  4  2 |  0  0  0  0   0  |
Nr. 4:  F  A  F  N  S  D  D  F  A  A  F  F  F  F  F  A  N  S  D  D  F  F  D  A  F  F  F  |  5  5  2 |  0  0  0  0   0  |
Nr. 5:  A  F  F  F  D  F  F  N  S  D  F  F  A  A  F  D  D  F  F  F  F  N  S  F  F  F  A  A  |  4  5  2 |  0  0  0  0   0  |
-------------------------------------------------------------------------------------------------------------------------
Nr. 6:  A  F  F  D  F  F  F  D  N  S  F  F  A  A  F  D  D  F  F  F  F  F  N  S  F  F  A  A  |  4  5  2 |  0  0  0  0   0  |
Nr. 7:  F  F  F  A  A  F  F  F  F  N  S  F  D  D  F  A  A  F  F  F  F  F  N  S  F  D  D  |  5  4  2 |  0  0  0  0   0  |
Nr. 8:  F  F  F  F  A  F  F  A  F  F  N  S  D  D  F  A  F  A  F  F  F  F  F  F  N  S  D  D  |  4  4  2 |  0  0  0  0   0  |
Nr. 9:  F  F  D  A  N  S  F  A  F  F  D  F  F  F  A  A  F  D  A  F  F  D  F  F  D  N  S  F  |  5  5  2 |  0  0  0  0   0  |
Nr.10:  D  D  A  F  F  F  F  D  F  F  A  N  S  F  F  F  F  D  A  F  F  A  F  F  D  F  N  S  |  5  4  2 |  0  1  0  0   1  |
-------------------------------------------------------------------------------------------------------------------------
Nr.11:  F  F  D  D  F  F  F  F  F  F  A  A  N  N  S  F  F  F  D  F  F  D  F  A  A  A  F  F  |  4  5  2 |  0  0  0  0   0  |
Nr.12:  D  D  A  F  F  F  F  F  F  F  D  A  F  F  F  F  F  F  N  N  N  S  A  F  F  F  F  F  |  3  3  3 |  0  0  0  0   0  |
Nr.13:  F  A  F  F  F  N  N  S  F  F  F  D  F  F  D  F  F  F  F  F  F  F  D  A  F  A  F  N  |  3  3  3 |  0  1  0  0   1  |
-------------------------------------------------------------------------------------------------------------------------
AV: 'Antal vagter'
WS: 'Weekendsammenlægning'
FØ: 'Fridagsønsker'
EV: 'Efterfølgende vagter'

Samlede pris: 2
Strategi: step
Benyttet tid: 1001.02


Yes (1001.02s cpu)
[eclipse 44]:
```

C.2 Gams

For overskuelighedens skyld er gamsprogrammet delt op i flere filer.

C.2.1 `dats.gms`

```
Sets
 d dage / M1, Ti1, O1, To1, F1, L1, S1,
          M2, Ti2, O2, To2, F2, L2, S2,
          M3, Ti3, O3, To3, F3, L3, S3,
          M4, Ti4, O4, To4, F4, L4, S4/

 dmlu1(d) /M1,Ti1,O1,To1,F1,L1/
 dmlu2(d) /M2,Ti2,O2,To2,F2,L2/
 dmfu2(d) /M2,Ti2,O2,To2,F2/
 dmlu3(d) /M3,Ti3,O3,To3,F3,L3/
 dmfu3(d) /M3,Ti3,O3,To3,F3/
 dmlu4(d) /M4,Ti4,O4,To4,F4,L4/
 dmfu4(d) /M4,Ti4,O4,To4,F4/


 t vagttyper / dag, aft, nat, sov, fri /
 a(t) arbejdstyper /dag, aft, nat/

 alias(t,tt);

```

C.2.2 `sim10.gms`

```
* sim10-1
* 13 medarbejdere  alle 20  timer,  ingen fridags onsker, alle nat
* behov 1 1 1

Sets
 s sygep / n1*n10 / ;

 alias(s,ss);


Parameter
 n(s) natkvalifikation
   / n1 1, n2 1, n3 1, n4 1, n5 1,
     n6 1, n7 1, n8 1, n9 1, n10 1/


ti(s) timetal
   /n1 20, n2 20, n3 20, n4 20, n5 20,
    n6 20, n7 20, n8 20, n9 20, n10 20/;

*behov pr dag
Table b(d,a)
                dag     aft     nat
M1              1       1       1
Ti1             1       1       1
O1              1       1       1
To1             1       1       1
F1              1       1       1
L1              1       1       1
S1              1       1       1
M2              1       1       1
Ti2             1       1       1
O2              1       1       1
To2             1       1       1
F2              1       1       1
L2              1       1       1
S2              1       1       1
M3              1       1       1
Ti3             1       1       1
O3              1       1       1
To3             1       1       1
F3              1       1       1
L3              1       1       1
```

```
S3              1       1       1
M4              1       1       1
Ti4             1       1       1
O4              1       1       1
To4             1       1       1
F4              1       1       1
L4              1       1       1
S4              1       1       1 ;

*fridagsonsker for medarbejderne
 Table  onsker(d,s)
                n1 n2 n3 n4 n5 n6 n7 n8 n9 n10
 M1             0  0  0  0  0  0  0  0  0  0
 Ti1            0  0  0  0  0  0  0  0  0  0
 O1             0  0  0  0  0  0  0  0  0  0
 To1            0  0  0  0  0  0  0  0  0  0
 F1             0  0  0  0  0  0  0  0  0  0
 L1             0  0  0  0  0  0  0  0  0  0
 S1             0  0  0  0  0  0  0  0  0  0
 M2             0  0  0  0  0  0  0  0  0  0
 Ti2            0  0  0  0  0  0  0  0  0  0
 O2             0  0  0  0  0  0  0  0  0  0
 To2            0  0  0  0  0  0  0  0  0  0
 F2             0  0  0  0  0  0  0  0  0  0
 L2             0  0  0  0  0  0  0  0  0  0
 S2             0  0  0  0  0  0  0  0  0  0
 M3             0  0  0  0  0  0  0  0  0  0
 Ti3            0  0  0  0  0  0  0  0  0  0
 O3             0  0  0  0  0  0  0  0  0  0
 To3            0  0  0  0  0  0  0  0  0  0
 F3             0  0  0  0  0  0  0  0  0  0
 L3             0  0  0  0  0  0  0  0  0  0
 S3             0  0  0  0  0  0  0  0  0  0
 M4             0  0  0  0  0  0  0  0  0  0
 Ti4            0  0  0  0  0  0  0  0  0  0
 O4             0  0  0  0  0  0  0  0  0  0
 To4            0  0  0  0  0  0  0  0  0  0
 F4             0  0  0  0  0  0  0  0  0  0
 L4             0  0  0  0  0  0  0  0  0  0
 S4             0  0  0  0  0  0  0  0  0  0 ;
```

C.2.3 variablenames.gms

```
Variables

*tildelingsvariabel
        x(d,t,s)  vagtvar for sygep

*fridognsvariable binaere
* d-dag a-aft n-nat s-sov f-fri
        ff(d,s)  lang fridognsperiode ff
        dfn(d,s) lang fridognsperiode dfn
        dfd(d,s) kort fridognsperiode dfd
        dfa(d,s) kort fridognsperiode dfa
        afa(d,s) kort fridognsperiode afa
        afn(d,s) kort fridognsperiode afn
        sfa(d,s) kort fridognsperiode sfa
        sfn(d,s) kort fridognsperiode sfn

*fridognsperioder overholdes over forste uge, binaere
        u1ff(s)  ff forste uge
        u1dfn(s) dfn forste uge
        u1dfd(s) dfd forste uge
        u1dfa(s) dfa forste uge
        u1afa(s) afa forste uge
        u1afn(s) afn forste uge
        u1sfa(s) sfa forste uge
        u1sfn(s) sfn forste uge

        su1fam35(s), su1fnm35(s) specialtilfaelde man 35 timer
        su1dfs35(s), su1dfs55(s) specialtilfaelde son 35 og 55 timer

*fridognsperioder overholdes over anden uge
        u2ff(s)  ff anden uge,
        u2dfn(s) dfn anden uge
        u2dfd(s) dfd anden uge
        u2dfa(s) dfa anden uge
```

```
         u2afa(s) afa anden uge
         u2afn(s) afn anden uge
         u2sfa(s) sfa anden uge
         u2sfn(s) sfn anden uge

         su2fam35(s), su2fnm35(s)
         su2dfm35(s), su2fnm55(s)
         su2dfs35(s), su2dfs55(s),

*fridognsperioder overholdes over tredje uge
         u3ff(s)  ff tredje uge
         u3dfn(s) dfn tredje uge
         u3dfd(s) dfd tredje uge
         u3dfa(s) dfa tredje uge
         u3afa(s) afa tredje uge
         u3afn(s) afn tredje uge
         u3sfa(s) sfa tredje uge
         u3sfn(s) sfn tredje uge

         su3fam35(s), su3fnm35(s)
         su3dfm35(s), su3fnm55(s)
         su3dfs35(s), su3dfs55(s)

*fridognsperioder overholdes over fjerde uge
         u4ff(s)  ff fjerde uge
         u4dfn(s) dfn fjerde uge
         u4dfd(s) dfd fjerde uge
         u4dfa(s) dfa fjerde uge
         u4afa(s) afa fjerde uge
         u4afn(s) afn fjerde uge
         u4sfa(s) sfa fjerde uge
         u4sfn(s) sfn fjerde uge

         su4fam35(s), su4fnm35(s)
         su4dfm35(s), su4fnm55(s)
         su4dfs35(s)

*fri hver anden weekend w-weeknr-tilfaelde binaer
         w11(s), w12(s)
         w22(s), w23(s)
         w33(s), w34(s)

*ovre og nedre graenser for D-dag A-aft
         DMinVar(s), DMaxVar(s)
         AMaxVar(s), AMinVar(s)
         NVO(s), NVN(s)

*lordag sondag ens binaere variable
         b1(t,s), b1sf(s)
         b2(t,s), b2sf(s)
         b3(t,s),b3sf(s)
         b4(t,s), b4sf(s)
*lordag sondag sovefri sum
         sf

*objektfunktionsvariable
         efc     efterfolgende vagter omkostning
         fric    fridagsonsker omkostning
         losoc   weekendsammenlaegning
         dvc     antal dagvagter
         avc     antal aftenvagter
         dvcmin  min antal dagvagter
         avcmin  min antal aftenvagter
         z       totale omkostning ;


Positive variables DMinVar,DMaxVar, AMaxVar,AMinVar,NVO,NVN;
Positive variables sf, efc,fric,dvc,avc;
Binary variable x;
Binary variable ff, dfn, dfd, dfa, afa, afn, sfa, sfn;
Binary variable u1ff, u1dfn, u1dfd, u1dfa, u1afa, u1afn,
          u1sfa, u1sfn,su1fam35, su1fnm35, su1dfs35, su1dfs55;
Binary variable u2ff, u2dfn, u2dfd, u2dfa, u2afa, u2afn,
         u2sfa,  u2sfn,su2fam35, su2fnm35, su2dfs35, su2dfs55;
Binary variable u3ff, u3dfn, u3dfd, u3dfa, u3afa, u3afn,
         u3sfa,  u3sfn,su3fam35, su3fnm35, su3dfs35, su3dfs55;
Binary variable u4ff, u4dfn, u4dfd, u4dfa, u4afa, u4afn,
         u4sfa,  u4sfn,su4fam35, su4fnm35, su4dfs35;
Binary variable w11,w12,w22,w23,w33,w34;
Binary variable b1,b1sf,b2,b2sf,b3,b3sf,b4,b4sf;
```

115

C.2.4 pars.gms

```
*laengder paa vagterne
Parameter Td;
 Td = 8;
Parameter Te;
 Te = 9.5;
Parameter Tn;
 Tn = 9.75;

*priser paa overtraedelserne
*fridagsonsker
Parameter P1;
 P1 =1;
*ens weekender
Parameter P2;
 P2 =1;
*sove-nat
Parameter P3;
 P3 =1;
*for mange/faa dagvagter
Parameter P4;
 P4 =1;


*Parametre til beregning af fordeling af vagter
*Behov for dag aften og nattevagter
Parameter Dns;
        Dns = sum(d,b(d,'dag'));
Parameter Ens;
        Ens = sum(d,b(d,'aft'));
Parameter Nns;
        Nns = sum(d,b(d,'nat'));


*Ressourcer til nattevagter
Parameter NatRes;
        NatRes = sum(s, ti(s)*n(s));
Parameter NatDel(s);
        NatDel(s) = (ti(s)*n(s)) / NatRes;
Parameter NatMax(s);
        NatMax(s) = ceil(NatDel(s)*Nns);
Parameter NatMin(s);
        NatMin(s) = n(s)*(NatMax(s)-1);


Parameter NewTi(s);
        NewTi(s) = 4*Ti(s) - (Tn*n(s)*NatDel(s)*Nns);

Parameter DAD(s);
        DAD(s) = NewTi(s) / sum(ss, NewTi(ss))

Parameter DaMa(s);
        DaMa(s) = ceil(DAD(s)*Dns);
Parameter DaMi(s);
        DaMi(s) = ceil(DAD(s)*Dns)-1;

Parameter AfMa(s);
        AfMa(s) = ceil(DAD(s)*Ens);
Parameter AfMi(s);
        AfMi(s) = ceil(DAD(s)*Ens)-1;

Parameter DagMax(s);
        DagMax(s) = DaMa(s);
Parameter DagMin(s);
        DagMin(s) = DaMi(s);
Parameter AftMax(s);
        AftMax(s) = AfMa(s);
Parameter AftMin(s);
        AftMin(s) = AfMi(s);
```

C.2.5 equationnames.gms

```
Equations


*       testeq(s,d,t)
* haarde
*       loso1e(s)
*       loso2e(s)
*       loso3e(s)
*       loso4e(s)

        NatOevgr(s), NatNedgr(s)        ovre og nedre graenser
        bop(d,a)        behovet opfyldes
        envgt(d,s)      envagt pr dag
        natkval(d,s)    nattekavlifikation
        sovnat(d,s)     sovedag kun efter nattevagt
        nateft(d,s)     nattevagt kun efterfoelges af nat sov
        dagaft(d,s)     ingen dagvagt efter aftenvagt
        w121(s), w122(s), w12v(s)       fri weekend et eller to
        w232(s), w233(s), w23v(s)       fri weekend to eller tre
        w343(s), w344(s), w34v(s)       fri weekend tre eller fire
        friprper(s,d)   fri min en dag pr syv dage

        frifri(d,s), dagfrinat(d,s)      lange fridognsperioder
        dagfridag(d,s), dagfriaften(d,s) korte fridognsperioder
        aftfriaft(d,s), aftfrinat(d,s)   korte fridognsperioder
        sovfriaften(d,s), sovfrinat(d,s) korte fridognsperioder

*       *fridognsperioder pr uge
        s11(s), s12(s)
        u1frifri(s), u1dafrna(s)
        u1dafrda(s), u1dafraf(s)
        u1affraf(s), u1affrna(s)
        u1sofraf(s), u1sofrna(s)
        u1f(s)

        u2frifri(s), u2dafrna(s)
        u2dafrda(s), u2dafraf(s)
        u2affraf(s), u2affrna(s)
        u2sofraf(s), u2sofrna(s)
        u2f(s)

        u3frifri(s), u3dafrna(s)
        u3dafrda(s), u3dafraf(s)
        u3affraf(s), u3affrna(s)
        u3sofraf(s), u3sofrna(s)
        u3f(s)

        u4frifri(s), u4dafrna(s)
        u4dafrda(s), u4dafraf(s)
        u4affraf(s), u4affrna(s)
        u4sofraf(s), u4sofrna(s)
        u4f(s), s4(s)

*bloede
        efcost(d,s)     uhensigtsmaessig vagt
        friocost        fridagsoensker

        loso1(s,t,tt)   lordag lig sondag
*       lososf1(s)      sovefrigatis
        loso2(s,t,tt)
*  lososf2(s)
        loso3(s,t,tt)
*  lososf3(s)
        loso4(s,t,tt)
*  lososf4(s)
*       lososf,
        lososcost

        DagOevgr(s)
        DagNedgr(s)
        dvcost
        dvcostmin
        AftOevgr(s)
        AftNedgr(s)
        avcost
        avcostmin
```

```
        cost            objektfunktion ;
```

C.2.6 vpgg.gms

```
$include dats
$include sim10
$include variablenames
$include pars
$include equationnames


* haarde begraensninger

*behovet skal vaere opfyldt
bop(d,a)..  sum(s, x(d,a,s)) =e= b(d,a) ;

*en vagt pr. dag
envgt(d,s) ..  sum(t, x(d,t,s)) =e= 1 ;

*kun nattevagt hvis man maa
natkval(d,s) ..  x(d,'nat',s) =l= n(s) ;

*sovedag kun efter nat
sovnat(d,s)$(ord(d)<28)..  x(d,'nat',s) =g= x(d+1,'sov',s) ;

*efter nat kun nat eller sovedag
*sidste dag kan man godt have nattevagt
nateft(d,s)$(ord(d)<28)..  x(d,'nat',s) =l= x(d+1,'nat',s) +
                                            x(d+1,'sov',s) ;

*ej aftenvagt efter dagvagt.
dagaft(d,s)$(ord(d)<28)..  x(d,'aft',s) + x(d+1,'dag',s) =l= 1 ;

*Fri hver anden weekend
w121(s)..       x('L1','fri',s) + x('S1','fri',s) =g= 2*w11(s);
w122(s)..       x('L2','fri',s) + x('S2','fri',s) =g= 2*w12(s);
w12v(s)..       w11(s) + w12(s) =g=1;

w232(s)..       x('L2','fri',s) + x('S2','fri',s) =g= 2*w22(s);
w233(s)..       x('L3','fri',s) + x('S3','fri',s) =g= 2*w23(s);
w23v(s)..       w22(s) + w23(s) =g=1;

w343(s)..       x('L3','fri',s) + x('S3','fri',s) =g= 2*w33(s);
w344(s)..       x('L4','fri',s) + x('S4','fri',s) =g= 2*w34(s);
w34v(s)..       w33(s) + w34(s) =g=1;

*mindst en fridag pr. lobende syv dage
friprper(s,d)$(ord(d)<23)..
        (x(d,'fri',s) + x(d+1,'fri',s) + x(d+2,'fri',s) +
         x(d+3,'fri',s) + x(d+4,'fri',s) + x(d+5,'fri',s) +
         x(d+6,'fri',s)) =g= 1;

*lange fridognsperioder
frifri(d,s)$(ord(d)<28)..
        x(d,'fri',s) + x(d+1,'fri',s) =g= 2*ff(d,s) ;
dagfrinat(d,s)$(ord(d)<27)..
        x(d,'dag',s)+x(d+1,'fri',s)+x(d+2,'nat',s) =g= 3*dfn(d,s) ;

*korte fridogn
dagfridag(d,s)$(ord(d)<27)..
        x(d,'dag',s)+x(d+1,'fri',s)+x(d+2,'nat',s) =g= 2*dfd(d,s) ;

dagfriaften(d,s)$(ord(d)<27)..
        x(d,'dag',s)+x(d+1,'fri',s)+x(d+2,'aft',s) =g= 2*dfa(d,s);

aftfriaft(d,s)$(ord(d)<27)..
        x(d,'aft',s)+x(d+1,'fri',s)+x(d+2,'aft',s) =g= 2*afa(d,s);

aftfrinat(d,s)$(ord(d)<27)..
        x(d,'aft',s)+x(d+1,'fri',s)+x(d+2,'nat',s) =g= 2*afn(d,s) ;

sovfriaften(d,s)$(ord(d)<27)..
        x(d,'sov',s)+x(d+1,'fri',s)+x(d+2,'aft',s) =g= 2*sfa(d,s);

sovfrinat(d,s)$(ord(d)<27)..
        x(d,'sov',s)+x(d+1,'fri',s)+x(d+2,'nat',s) =g= 2*sfn(d,s) ;
```

```
*fridognsperioder overholdt i ugerne
$include friuge1
$include friuge2
$include friuge3
$include friuge4

*ovre og nedre graense for nattevagter
NatOevgr(s)..  sum(d,x(d,'nat',s))  =l= n(s)*NatMax(s);
NatNedgr(s)..  sum(d,x(d,'nat',s))  =g= n(s)*(NatMax(s)-1);


**************************************
* bloede begraensninger

*uhensigtsmaessige vagter
efcost(d,s)..  efc =g= x(d,'sov',s) + x(d+1,'nat',s)-1;

*fridagsonsker
friocost..  fric =e= sum((s,d),
       onsker(d,s) * (x(d,'dag',s) + x(d,'nat',s) +
                       x(d,'aft',s)+x(d,'sov',s)));

*lordag sondag skal helst vaere ens
loso1(s,t,tt)$(ord(t) eq ord(tt))..
       (x('L1',t,s) + x('S1',tt,s))  =g= 2*b1(t,s);
* dog sove-fri gratis
*lososf1(s)..  (x('L1','sov',s) + x('S1','fri',s))  =g= 2*b1sf(s);
*loso1e(s).. sum(t, b1(t,s)) + b1sf(s) =l= 1;

loso2(s,t,tt)$(ord(t) eq ord(tt))..
       (x('L2',t,s) + x('S2',tt,s))  =g= 2*b2(t,s);
*lososf2(s)..  (x('L2','sov',s) + x('S2','fri',s))  =g= 2*b2sf(s);
*loso2e(s).. sum(t, b2(t,s)) + b2sf(s) =l= 1;

loso3(s,t,tt)$(ord(t) eq ord(tt))..
       (x('L3',t,s) + x('S3',tt,s))  =g= 2*b3(t,s);
*lososf3(s)..  (x('L3','sov',s) + x('S3','fri',s))  =g= 2*b3sf(s);
*loso3e(s).. sum(t, b3(t,s)) + b3sf(s) =l= 1;

loso4(s,t,tt)$(ord(t) eq ord(tt))..
       (x('L4',t,s) + x('S4',tt,s))  =g= 2*b4(t,s);
*lososf4(s)..  (x('L4','sov',s) + x('S4','fri',s))  =g= 2*b4sf(s);
*loso4e(s).. sum(t, b4(t,s)) + b4sf(s) =l= 1;

*lososf..  sf =e= sum(s, b1sf(s) + b2sf(s) + b3sf(s) + b4sf(s));
losocost..  losoc =e= (0 - sum((t,s),b1(t,s)+b2(t,s)+b3(t,s)+b4(t,s)));


* dagvagter deles ligeligt
DagOevgr(s)..   DagMax(s) + DMaxVar(s) =g= sum(d,x(d,'dag',s));
DagNedgr(s)..   sum(d,x(d,'dag',s)) + DMinVar(s) =g= DagMin(s);
*dvcost..       dvc =e= sum(s,(DMaxVar(s) + DminVar(s)));
dvcost..        dvc =e= sum(s,DMaxVar(s));
dvcostmin..     dvcmin =e= sum(s,DMinVar(s));

* aftenvagter deles ligeligt
AftOevgr(s)..   AftMax(s) + AMaxVar(s) =g= sum(d,x(d,'aft',s));
AftNedgr(s)..   sum(d,x(d,'aft',s)) + AMinVar(s) =g= AftMin(s);
avcost..        avc =e= sum(s,AMaxVar(s));
*avcost..       avc =e= sum(s,(AMaxVar(s) + AMinVar(s)));
avcostmin..     avcmin =e= sum(s,AMinVar(s));

* objektfunktion
cost..          z =e= P1*fric + P2*losoc +
                      P3*efc + P4*(dvc+avc) + P4*(dvcmin+avcmin);


Model vagtplan /all/ ;
solve vagtplan using mip minizing z ;
display efc.l;
display fric.l;
display dvc.l;
display avc.l;
display losoc.l;
*display DaMa('n1').l;
```

C.2.7 friuge1.gms

```
*fridognsperioder paa forste uge

s11(s).. x('M1','fri',s) + x('Ti1','aft',s) =g= 2*su1fam35(s);
s12(s).. x('M1','fri',s) + x('Ti1','nat',s) =g= 2*su1fnm35(s);


u1frifri(s)..   u1ff(s) =e= sum(dmlu1, ff(dmlu1,s));
u1dafrna(s)..   u1dfn(s) =e= sum(dmlu1, dfn(dmlu1,s));

u1dafrda(s)..   u1dfd(s) =e= sum(dmlu1,dfd(dmlu1,s));
u1dafraf(s)..   u1dfa(s) =e= sum(dmlu1,dfa(dmlu1,s));
u1affraf(s)..   u1afa(s) =e= sum(dmlu1,afa(dmlu1,s));
u1affrna(s)..   u1afn(s) =e= sum(dmlu1,afn(dmlu1,s));
u1sofraf(s)..   u1sfa(s) =e= sum(dmlu1,sfa(dmlu1,s));
u1sofrna(s)..   u1sfn(s) =e= sum(dmlu1,sfn(dmlu1,s));


*opfyldelse af fridognsperioder uge1
u1f(s)..        2*u1ff(s) + 2*u1dfn(s) + u1dfd(s) +
                  u1dfa(s) + u1afa(s) + u1afn(s) +
                  u1sfa(s) + u1sfn(s)  =g= 2;
```

C.2.8 friuge2.gms

```
*fridognsperioder paa anden uge

u2frifri(s)..   u2ff(s) =e= sum(dmlu2, ff(dmlu2,s));
u2dafrna(s)..   u2dfn(s) =e= sum(dmfu2, dfn(dmfu2,s));

u2dafrda(s)..   u2dfd(s) =e= sum(dmfu2,dfd(dmfu2,s));
u2dafraf(s)..   u2dfa(s) =e= sum(dmfu2,dfa(dmfu2,s));
u2affraf(s)..   u2afa(s) =e= sum(dmfu2,afa(dmfu2,s));
u2affrna(s)..   u2afn(s) =e= sum(dmfu2,afn(dmfu2,s));
u2sofraf(s)..   u2sfa(s) =e= sum(dmfu2,sfa(dmfu2,s));
u2sofrna(s)..   u2sfn(s) =e= sum(dmfu2,sfn(dmfu2,s));

*opfyldelse af fridognsperioder uge2
u2f(s)..        2*u2ff(s) + 2*u2dfn(s) + u2dfd(s) +
                  u2dfa(s) + u2afa(s) + u2afn(s) +
                  u2sfa(s) + u2sfn(s)  =g= 2;
```

C.2.9 friuge3.gms

```
*fridognsperioder paa tredje uge

u3frifri(s)..   u3ff(s) =e= sum(dmlu3, ff(dmlu3,s));
u3dafrna(s)..   u3dfn(s) =e= sum(dmfu3, dfn(dmfu3,s));

u3dafrda(s)..   u3dfd(s) =e= sum(dmfu3,dfd(dmfu3,s));
u3dafraf(s)..   u3dfa(s) =e= sum(dmfu3,dfa(dmfu3,s));
u3affraf(s)..   u3afa(s) =e= sum(dmfu3,afa(dmfu3,s));
u3affrna(s)..   u3afn(s) =e= sum(dmfu3,afn(dmfu3,s));
u3sofraf(s)..   u3sfa(s) =e= sum(dmfu3,sfa(dmfu3,s));
u3sofrna(s)..   u3sfn(s) =e= sum(dmfu3,sfn(dmfu3,s));


*opfyldelse af fridognsperioder uge3
u3f(s)..        2*u3ff(s) + 2*u3dfn(s) + u3dfd(s) +
                  u3dfa(s) + u3afa(s) + u3afn(s) +
                  u3sfa(s) + u3sfn(s)  =g= 2;
```

C.2.10 friuge4.gms

```
*fridognsperioder paa fjerde uge
```

```
u4frifri(s)..   u4ff(s) =e= sum(dmlu4, ff(dmlu4,s));
u4dafrna(s)..   u4dfn(s) =e= sum(dmfu4, dfn(dmfu4,s));

u4dafrda(s)..   u4dfd(s) =e= sum(dmfu4,dfd(dmfu4,s));
u4dafraf(s)..   u4dfa(s) =e= sum(dmfu4,dfa(dmfu4,s));
u4affraf(s)..   u4afa(s) =e= sum(dmfu4,afa(dmfu4,s));
u4affrna(s)..   u4afn(s) =e= sum(dmfu4,afn(dmfu4,s));
u4sofraf(s)..   u4sfa(s) =e= sum(dmfu4,sfa(dmfu4,s));
u4sofrna(s)..   u4sfn(s) =e= sum(dmfu4,sfn(dmfu4,s));

s4(s).. x('L4','dag',s) + x('S4','fri',s) =g= 2*su4dfs35(s);

*opfyldelse af fridognsperioder uge4
u4f(s)..        2*u4ff(s) + 2*u4dfn(s) + u4dfd(s) +
                  u4dfa(s) + u4afa(s) + u4afn(s) +
                  u4sfa(s) + u4sfn(s) + su4dfs35(s)  =g= 2;
```

Bilag D

Testkørsler

D.1 hil18

GAMS

```
Resource limit ecxeeded, no integer solution exists
```

vp(hil18,hil18,28,0,0,step,1000,0,2000).

```
no
```

vp(hil18,hil18,14,2,0,step,500,0,2000).

```
no
```

vp(hil18,hil18,28,0,1,step,1000,0,2000).

```
no
```

vp(hil18,hil18,14,2,1,step,500,0,2000).

```
no
```

D.2 hil20

GAMS

```
Resource limit ecxeeded, no integer solution exists
```

vp(hil20,hil20,28,0,0,step,1000,0,2000).

```
Benyttet tid: 297.78
Found a solution with cost 24
Branch-and-bound timeout!

      m t o t f l s m t o t f l s m t o t f l s m t o t f l s | D A N | AV WS FØ EV sum |
-----------------------------------------------------------|-------|-----------------|
Nr. 1: N S F F F N N S D D D F F F D D F F F A A A A A A F F F | 5 6 3 | 0  0  0  0  0   |
Nr. 2: N S F D F N N S D D D F F F D D F F F A A A A A A F F F | 6 6 3 | 0  0  0  0  0   |
Nr. 3: F N S F D F F D A A F F N N S A A F D F F D D D F F A A | 6 6 3 | 0  0  0  0  0   |
Nr. 4: F F A F F F F D A F F N N N S A A F D F F D D D F F A A | 5 6 3 | 0  0  0  0  0   |
Nr. 5: F F F F F A A F D D D D F F F D F F D A A A A A A F A F F | 6 8 0 | 0  0  0  0  0   |
-----------------------------------------------------------|-------|-----------------|
Nr. 6: F A F F F A A F A F F D F F F F D F F N N S D F D F F F | 4 4 2 | 1  0  0  0  1   |
Nr. 7: F F D D D A A F A A A F F F F F D D N N N S F F F F F F | 5 5 3 | 0  0  0  0  0   |
Nr. 8: F F D D D D D F F A A A F F F A A F F F F F F F A A N N | 5 7 2 | 1  0  0  0  1   |
Nr. 9: F F F F F F D D D F F A A F F D A A A A A F F F F F F D N N | 5 6 2 | 0  0  0  0  0   |
Nr.10: D N S F F D D N S F A A F F A F F A F D D A F F F F F F | 5 5 2 | 0  0  0  0  0   |
-----------------------------------------------------------|-------|-----------------|
Nr.11: F D N N S F F N S F F D A A A F D D F F F D F F F F A A | 5 5 3 | 0  0  0  0  0   |
Nr.12: F D N N S F F A N S F F A A A F F A F F F F F F D D D D | 5 5 3 | 0  0  0  0  0   |
Nr.13: F D F F A F F A N S F F A A N N S A F F F F F F F D D D D | 5 5 3 | 0  0  0  0  0   |
Nr.14: D A F F A F F F F N N S D D A F F F A F F F A A F F D D | 5 6 2 | 0  0  0  0  0   |
Nr.15: A F F A A F F F F N N S D D F F F F A F F F F F F F F F | 2 4 2 | 0  0  0  0  0   |
-----------------------------------------------------------|-------|-----------------|
Nr.16: A F F A A F F A F A F F D D N N N S F F F F F D F N S F | 3 5 4 | 0  0  0  0  0   |
Nr.17: D F D A N S F A F F F A F F F F N N S D D F N S A A F F | 4 5 4 | 0  0  0  0  0   |
Nr.18: F F A A N S F F F F F F F F F F F N S D D N S F N S F F | 2 2 4 | 3  0  0  0  3   |
Nr.19: A A A F F F F F F F F N S F F F F D A F F F F N N N S F | 1 4 4 | 2  0  0  0  2   |
Nr.20: A A A F F F F F F F F F F F F F F N S F N N N S A F F | 0 4 4 | 3  0  0  0  3   |
-----------------------------------------------------------|-------|-----------------|
Samlede pris: 10
```

vp(hil20,hil20,14,2,0,step,500,0,2000).

```
no

(Delløsning: pris: 1, tid: 7).
```

vp(hil20,hil20,28,0,1,step,1000,0,2000).

```
Benyttet tid: 4.85000000000036
Found a solution with cost 36
Branch-and-bound timeout!

      m t o t f l s m t o t f l s m t o t f l s m t o t f l s | D A N | AV WS FØ EV sum |
-----------------------------------------------------------|-------|-----------------|
Nr. 1: F N S D F D D N N S A F F F A A A F F D D A A F F F F F | 5 6 3 | 0  0  0  0  0   |
Nr. 2: D D D D A F F A F F N S D D N S F A A F F A A F F N S F | 6 6 3 | 0  0  0  0  0   |
Nr. 3: F F A A N S F F D D D A F F A N S F F D D N S A A F F F | 5 6 3 | 0  0  0  0  0   |
Nr. 4: D D F N S F F A F N S N S F A F A A A F F D D D D D F F | 7 5 3 | 2  0  0  1  3   |
Nr. 5: F A F F F F F F A A A F F D D F A A A A A F F F F D D D D D | 7 8 0 | 1  0  0  0  1   |
-----------------------------------------------------------|-------|-----------------|
Nr. 6: A A F F F D D F F F F F F F F D F N N S A A A F F D A F F | 4 6 2 | 3  0  0  0  3   |
Nr. 7: N S D A A F F F F F F F D A A F F F F D F F F D F F A N N | 4 5 3 | 0  0  0  0  0   |
Nr. 8: F F A F D A A F F F F F F F F D F F F F A A F N S N S F F | 2 5 2 | 2  0  0  1  3   |
Nr. 9: A A F F F F F F D A A F F N N S A F D D F F D F F F A A A | 4 8 2 | 2  0  0  0  2   |
Nr.10: A A A F F F F F F F N N S D D F F F F F F F F F F F D D D | 5 3 2 | 2  0  0  0  2   |
-----------------------------------------------------------|-------|-----------------|
Nr.11: A F F F F A A A A A F D F F D D D F F F F F F F F N N N | 4 6 3 | 0  0  0  0  0   |
Nr.12: N S F F F F F F D F F A A N N S A F D D F F D F F F F A A | 4 5 3 | 0  0  0  0  0   |
Nr.13: F F N S F N N S F D A F F F F D F F F D D A A A A F F F | 4 5 3 | 0  0  0  0  0   |
Nr.14: F F N S A F F F D F F N S F A F D D A F F F D A A A F F | 4 6 2 | 0  0  0  0  0   |
Nr.15: F F F A A N N S F F D A F F F D F F F F F F F F F F F F | 2 3 2 | 0  0  0  0  0   |
-----------------------------------------------------------|-------|-----------------|
Nr.16: D D D F D F F F F F F F A A F F N N S F F F F N N S A A | 4 4 4 | 0  0  0  0  0   |
Nr.17: F F F F D N S F D D D A A F F F F A A N N N S F A F F F F | 4 5 4 | 0  0  0  0  0   |
Nr.18: F F A N S F F N N S F F A A N S F F F F F F F D A F D D | 3 4 4 | 0  0  0  0  0   |
Nr.19: F N S A F A A F F F D D F F F F D F N N N S A F F F F F | 3 4 4 | 0  0  0  0  0   |
Nr.20: F F F F D D D F A A F F F F F N S F F A A N N N S F F F | 3 4 4 | 0  0  0  0  0   |
-----------------------------------------------------------|-------|-----------------|
Samlede pris: 14
```

vp(hil20,hil20,14,2,1,step,500,0,2000).

```
no

(Delløsning: pris: 8, tid: 12).
```

D.3 hol13

GAMS

```
Resource limit ecxeeded, no integer solution exists
%
% Uden Amin,Dmin
%
%      m t o t f l s m t o t f l s m t o t f l s m t o t f l s | D A N |
%-----------------------------------------------------------------|-------|
%Nr. 1: F A N N S F F F F A F F D D F F A F A F F D F A F F D D | 5 5 2 |
%Nr. 2: F N S F F D D F F A F D F F F N S F A A A A F D F F F F | 4 5 2 |
%Nr. 3: F A F F F F F F D F F D N N S D F A F F F F F A F F A A | 3 5 2 |
%Nr. 4: F F D F A F F D F F F N S F D F F D D F F F F F A F N S | 5 2 2 |
%Nr. 5: A F F D F N N S D F F F F F F F D F F D D A F F A A F F | 5 4 2 |
%-----------------------------------------------------------------|-------|
%Nr. 6: D F A F F F F F A F F A F F A F A F F N S F N S F D F F | 2 5 2 |
%Nr. 7: F D D A N S A F N S A F F F F F F F F D D D A N S F F F | 5 4 3 |
%Nr. 8: F F F A F D D F A F A F F F D F F A N S F N S F F A F F | 3 5 2 |
%Nr. 9: D F A F F A F A F D D F F F N S F N S A A F D F F D F F | 5 5 2 |
%Nr.10: N S F F D F F F F F F A A A F A F D F F F F F D D F D N | 5 4 2 |
%-----------------------------------------------------------------|-------|
%Nr.11: A F F F F F F A F D F F D D F D N S F F F F A F F N S D | 5 3 2 |
%Nr.12: F D F F D A A N S F D F F F A A F F D F N S F F D F F F | 5 4 2 |
%Nr.13: S F F D A F F D F N N S A A F F D F F F F F F D F N S A A | 4 5 3 |
%
%Samlede pris: 9
```

vp(hol13,hol13,28,0,0,step,1000,0,2000).

```
Benyttet tid: 22.6500000000005
Found a solution with cost 10
Branch-and-bound timeout!

      m t o t f l s m t o t f l s m t o t f l s m t o t f l s | D A N | AV WS FØ EV sum |
-----------------------------------------------------------------|-------|-----------------|
Nr. 1: F F F D F N N S D D D F F F F D F F F A A A A F F F F F | 5 4 2 | 0  0  0  0  0   |
Nr. 2: A F F F F F F D A A F F N N S D D D F F F F D F F F A A | 5 5 2 | 0  0  0  0  0   |
Nr. 3: A F F F F A A F D D D D F F F A A F F N N S F D F F F F | 5 5 2 | 0  0  0  0  0   |
Nr. 4: F F F F F A A F A A F D F F F F D D D F F F F D F F N N | 5 4 2 | 0  0  0  0  0   |
Nr. 5: N N S F F D D F F F F A F F D A F F D A A F F F D A F F | 5 5 2 | 0  0  0  0  0   |
-----------------------------------------------------------------|-------|-----------------|
Nr. 6: F F N N S D D F F F A A F F F F A A F D D F F F F A F F | 4 5 2 | 0  0  0  0  0   |
Nr. 7: F F D D D F F N N S F F A A F F F F F F F D D F F F A A | 5 4 2 | 0  0  0  0  0   |
Nr. 8: F A F F F F F F F N N S A A F F F A A F F F F F D D D D | 4 5 2 | 0  0  0  0  0   |
Nr. 9: F F F F F F F A F F A F D D N N S F A F F A F F F D D D | 5 4 2 | 0  0  0  0  0   |
Nr.10: F D F F D F F A F F F F D D A F N N S F F D A A A F F F | 5 5 2 | 0  0  0  0  0   |
-----------------------------------------------------------------|-------|-----------------|
Nr.11: F F D A N S F D F F F F F F A F F F F D D N S A A F F F | 4 4 2 | 0  0  0  0  0   |
Nr.12: D D A A A F F F F F F N S F D F F F F F F F F N S N S F | 3 3 3 | 0  0  0  1  1   |
Nr.13: D A A F A F F F F F F F F F F F F F N S F F N S N S F F | 1 3 3 | 2  0  0  1  3   |
-----------------------------------------------------------------|-------|-----------------|
Samlede pris: 4
```

vp(hol13,hol13,14,2,0,step,500,0,2000).

```
Benyttet tid: 564.78
Found a solution with cost 2
Branch-and-bound timeout!

      m t o t f l s m t o t f l s m t o t f l s m t o t f l s | D A N | AV WS FØ EV sum |
-----------------------------------------------------------------|-------|-----------------|
Nr. 1: N S F F D A A F D F F F F F N S F F D A A F F D F F F F | 4 4 2 | 0  0  0  0  0   |
Nr. 2: F N S F F A A F D D F D F F A N S F F A A F D F F D F F | 5 5 2 | 0  0  0  0  0   |
Nr. 3: F F N S F D D F A A F F F F F F N S F D D A A F F D F F | 5 4 2 | 0  0  0  0  0   |
Nr. 4: F A F N S D D F A A F F F F F F A N S D D F F D A F F F | 5 5 2 | 0  0  0  0  0   |
Nr. 5: A F F F D F F N S D F F A A F D D F F F F N S F F F A A | 4 5 2 | 0  0  0  0  0   |
-----------------------------------------------------------------|-------|-----------------|
Nr. 6: A F F D F F F D N S F F A A F D D F F F F F F N S F F A A | 4 5 2 | 0  0  0  0  0   |
Nr. 7: F F F A A F F F F N S F D D D F A A F F F F F F N S F D D | 5 4 2 | 0  0  0  0  0   |
Nr. 8: F F F F A F F A F F N S D D F A F A F F F F F F N S D D | 4 4 2 | 0  0  0  0  0   |
Nr. 9: F F D A N S F A F F D F F F A A F D A F F D F F D N S F | 5 5 2 | 0  0  0  0  0   |
Nr.10: D D A F F F F D F F A N S F F F F D A F F A F F D F N S | 5 4 2 | 0  1  0  0  1   |
-----------------------------------------------------------------|-------|-----------------|
Nr.11: F F D D F F F F F F A A N N S F F F D F F D F A A A F F | 4 5 2 | 0  0  0  0  0   |
Nr.12: D D A F F F F F F F F D A F F F F F F F N N N S A F F F F | 3 3 3 | 0  0  0  0  0   |
Nr.13: F A F F F N N S F F F D F F F D F F F F F F F F D A F A F N | 3 3 3 | 0  1  0  0  1   |
-----------------------------------------------------------------|-------|-----------------|
Samlede pris: 2
(Delløsning: pris 2, tid: 1)
```

vp(hol13,hol13,28,0,1,step,1000,0,2000).

```
Benyttet tid: 5.96000000000004
Found a solution with cost 6
Branch-and-bound timeout!

      m t o t f l s m t o t f l s m t o t f l s m t o t f l s | D A N | AV WS FØ EV sum |
-----------------------------------------------------------------|-------|-----------------|
Nr. 1: F F D D N S F N S F F F F F D F F F A F F A A A F D F F | 4 4 2 | 0  0  0  0   0  |
Nr. 2: F F A F D D D F D D F A F F F F F A F A A F N S N S F F | 5 5 2 | 0  0  0  1   1  |
Nr. 3: N N S F F F F F F F F D A A F F F D D F F D F F D A A A | 5 5 2 | 0  0  0  0   0  |
Nr. 4: D F F D D A A F F F N S F F N S F F F A A F D D F F F F | 5 4 2 | 0  0  0  0   0  |
Nr. 5: F D F A A F F D F F D D F F A F F D A F F D F N S N S F | 6 4 2 | 1  0  0  1   2  |
-----------------------------------------------------------------|-------|-----------------|
Nr. 6: F A F F F D D F F F A A F F A N S F F D D N S A A F F F | 4 6 2 | 1  0  0  0   1  |
Nr. 7: A F F F F F F A A A F F D D F A N N S F F F F F F D D D | 5 5 2 | 0  0  0  0   0  |
Nr. 8: A A A F F N N S F F F F F F F F F F D D F D D A A F F | 4 5 2 | 0  0  0  0   0  |
Nr. 9: F F N N S F F F D F D F A A F D D F F F F F F F F A A | 4 4 2 | 0  0  0  0   0  |
Nr.10: D D D A A F F F N N S F F F F F F F F F A A F D F F F | 4 4 2 | 0  0  0  0   0  |
-----------------------------------------------------------------|-------|-----------------|
Nr.11: F F F F F F F D A A F F D D F A A F D F F F F F F N N | 4 4 2 | 0  0  0  0   0  |
Nr.12: F F F F F F F F F D A N N N S F A A F F F F F F F D D | 3 3 3 | 0  0  0  0   0  |
Nr.13: F F F F F A A A F F F F F F D D D F N N N S F F F F F F | 3 3 3 | 0  0  0  0   0  |
-----------------------------------------------------------------|-------|-----------------|
Samlede pris: 4
```

vp(hol13,hol13,14,2,1,step,500,0,2000).

```
Benyttet tid: 504.58
Found a solution with cost 2
Branch-and-bound timeout!

      m t o t f l s m t o t f l s m t o t f l s m t o t f l s | D A N | AV WS FØ EV sum |
-----------------------------------------------------------------|-------|-----------------|
Nr. 1: F D F N S A A F F F F D F F F N S F D F F D F F A A F F | 4 4 2 | 0  0  0  0   0  |
Nr. 2: F F A F D A A F D F N S F F D F N S F A A F D D F F F F | 5 5 2 | 0  0  0  0   0  |
Nr. 3: N S F F F F F F F D D D A A A F F F D F F D N S F F A A | 5 5 2 | 0  0  0  0   0  |
Nr. 4: D D D F F F F D F N S F D D F F F A A F F A A F F N S F | 6 4 2 | 1  0  0  0   1  |
Nr. 5: F F D F F F F F F D A A F F F A A F F N N S D D D F F F | 5 4 2 | 0  0  0  0   0  |
-----------------------------------------------------------------|-------|-----------------|
Nr. 6: A N S F F F F F F F F D D A F F F A F F N S A F F D D | 4 4 2 | 0  0  0  0   0  |
Nr. 7: A A F A A F F F N S F F A A N S F F F F F F F F D D D D | 4 6 2 | 1  0  0  0   1  |
Nr. 8: F F F D D F F A A F F F N N S F D D F F F F F F F D A A | 5 4 2 | 0  0  0  0   0  |
Nr. 9: F F F D N S F D A A F F F F F F N S D D F F A A A F F | 4 5 2 | 0  0  0  0   0  |
Nr.10: F F N S F D D A F A F F F F F F A A F D D F F F N S F F | 4 4 2 | 0  0  0  0   0  |
-----------------------------------------------------------------|-------|-----------------|
Nr.11: F F F A F D D N S F A A F F D D F F F A A F F N S F F F | 4 5 2 | 0  0  0  0   0  |
Nr.12: D A A F F F F F F F D N S F F A F D F F F F F F F N N | 3 3 3 | 0  0  0  0   0  |
Nr.13: F F F F A N N S D F F F F F F D D F N S F A A F F F F F | 3 3 3 | 0  0  0  0   0  |
-----------------------------------------------------------------|-------|-----------------|
Samlede pris: 2
(delløsning: pris: 5, tid: 105)
```

D.4 hol13t2

GAMS

```
Resource limit ecxeeded, no integer solution exists
%
% Uden Amin, Dmin:
%
%      m t o t f l s m t o t f l s m t o t f l s m t o t f l s | D A N |
%-----------------------------------------------------------------|-------|
%Nr. 1: S F F A F A A F F A F F F F F D N S F D D D F F F F F F | 4 4 1 |
%Nr. 2: F N S F D F F A F F F D A A F A F D F F F F F F F F D F | 4 4 1 |
%Nr. 3: A F F F F F F F A F F F D D D F F F F F F F A F D F N N | 4 3 2 |
%Nr. 4: F D F N S F F F F F F F D D N S F F A F F A F F F D A A | 4 4 2 |
%Nr. 5: F F D F F N N S F F F F F F A F D A F D D F F A F A F F | 4 4 2 |
%-----------------------------------------------------------------|-------|
%Nr. 6: D F F F F D D F F A F F F F F F F F D F F F A A A N S D | 5 4 1 |
%Nr. 7: F F N S A F F D F D D F F F F A F D F F A F F F A F F F | 4 4 1 |
%Nr. 8: S F D D F F F D A F D F F F F F A A F A N S F F N S F F | 4 4 2 |
%Nr. 9: F A F F N S F F D F A A F F A F D F N N S F D F D D F F | 5 4 3 |
%Nr.10: D A A F F D D F N N S A F F D F A F D A A N S D F F F F | 6 6 3 |
%-----------------------------------------------------------------|-------|
%Nr.11: N S F F D A A F D F N S F F F D F N S F F F F D F F D D | 6 2 3 |
%Nr.12: S D A A F F F A F F A N N N S F F F A F F A N S F F A F | 1 7 4 |
%Nr.13: A F F D A F F N S D F D A A F N S F F F F D D N S A F A | 5 6 3 |
%
%Samlede pris: 9
```

vp(hol13,hol13t2,28,0,0,step,1000,0,2000).

```
Benyttet tid: 18.6800000000003
Found a solution with cost 7
Branch-and-bound timeout!

      m t o t f l s m t o t f l s m t o t f l s m t o t f l s | D A N | AV WS FØ EV sum |
-----------------------------------------------------------------|-------|-----------------|
Nr. 1: N S F F F A A F D D D F F F F F F F F A A F F F F F F F | 3 4 1 | 0  0  0  0   0  |
Nr. 2: F N S F F A A F D D D F F F F F F F D A A F D F F F F F | 5 4 1 | 1  0  0  0   1  |
Nr. 3: F F N S F D D F A A A F F F F F F F D D D F F A F F F F | 5 4 1 | 1  0  0  0   1  |
Nr. 4: F A F N S D D F F A A A F F F F F F F D D F F F F F F F | 4 4 1 | 0  0  0  0   0  |
Nr. 5: F F F F F N N S F F F D F F F D F F F F F F F D A F A A | 3 3 2 | 0  0  0  0   0  |
-----------------------------------------------------------------|-------|-----------------|
Nr. 6: F F F F F F F F F F D N N S D F F F F F F F D A F A A | 3 3 2 | 0  0  0  0   0  |
Nr. 7: A F F F F F F D F F F A F F F A F D F N N S F F D F F F | 3 3 2 | 0  0  0  0   0  |
Nr. 8: F D D D D F F N S F F F A A F A A A F F F D D F F F N N | 6 5 3 | 0  0  0  0   0  |
Nr. 9: D F F F D F F A N S F F A A N N S A F F F A F F D D D D | 6 5 3 | 0  0  0  0   0  |
Nr.10: A F F F A F F F F N N S D D A F N S A F F A A F F D D D | 5 6 3 | 0  0  0  0   0  |
-----------------------------------------------------------------|-------|-----------------|
Nr.11: F D D D A F F A A F F F D D D F A N S F F N N S F A F F | 6 5 3 | 0  0  0  0   0  |
Nr.12: F F A A N S F D F F F F F F A F D F N S F D A N S A F F | 3 5 3 | 2  0  0  0   2  |
Nr.13: D A A A F F F F F F F F N S F D F D D A F F F F A N N S F | 4 5 3 | 1  0  0  0   1  |
-----------------------------------------------------------------|-------|-----------------|
Samlede pris: 5
```

vp(hol13,hol13t2,14,2,0,step,500,0,2000).

```
Found a solution with cost 36
Branch-and-bound timeout!

      m t o t f l s m t o t f l s m t o t f l s m t o t f l s | D A N | AV WS FØ EV sum |
-----------------------------------------------------------------|-------|-----------------|
Nr. 1: F F F D D A A F D F F F F F N S F F F A A F F F F F F F | 3 4 1 | 0  0  0  0   0  |
Nr. 2: F A F F D A A F D F F D F F D N S F F A A F D D D F F F | 7 5 1 | 4  0  0  0   4  |
Nr. 3: A F F F F D D F A A F D F F F F N S F D D F D F F A F F | 6 4 1 | 2  0  0  0   2  |
Nr. 4: A F F F F D D F A A F F F F F D F N S D D F F F F F F F | 5 3 1 | 1  0  0  0   1  |
Nr. 5: F F F F F F F F D F D F F N N S D F F F F F F F F F A A | 3 2 2 | 0  0  0  0   0  |
-----------------------------------------------------------------|-------|-----------------|
Nr. 6: N S F F F F F F F D F F A A A F D F F F F N S F F F A A | 2 5 2 | 2  0  0  0   2  |
Nr. 7: F N S F A F F F F F F F A A F F F F F F F F N S F F D D | 2 3 2 | 0  0  0  0   0  |
Nr. 8: F F D A A N N S F F D F F F F A A F N S F A F D D D F F | 5 5 3 | 0  0  0  0   0  |
Nr. 9: F F D A F F F A F F D F D D F A A A F F F D F N S F N N | 5 5 3 | 0  0  0  0   0  |
Nr.10: F F N N S F F A F F A F D D F F D A A F F A F F N S D D | 5 5 3 | 0  0  0  0   0  |
-----------------------------------------------------------------|-------|-----------------|
Nr.11: F D F D N S F N S F A A F F D F F D A N S F A A F D F F | 5 5 3 | 0  1  0  0   1  |
Nr.12: D D A F F F F D N N S A F F F F F D D F N S A A A A F F | 5 6 3 | 0  1  0  0   1  |
Nr.13: D A A F F F F F F F N N S F A F F F D F F D F F A N S F | 3 4 3 | 3  0  0  0   3  |
-----------------------------------------------------------------|-------|-----------------|
Samlede pris: 14

(Delløsning: pris: 1, tid: 4)
```

vp(hol13,hol13t2,28,0,1,step,1000,0,2000).

```
Benyttet tid: 7.63999999999942
Found a solution with cost 13
Branch-and-bound timeout!

      m t o t f l s m t o t f l s m t o t f l s m t o t f l s | D A N | AV WS FØ EV sum |
-----------------------------------------------------------------|-------|-----------------|
Nr. 1: N S F F F D D A F F F F F F F F F F D D D A A F F F F F | 5 3 1 | 1  0  0  0   1  |
Nr. 2: F F A A F F F F F F A F D D D D A F D F F F F F F D N S F | 6 4 1 | 2  0  0  0   2  |
Nr. 3: D A F F F A A F F F D F F F F F F D D F A A F F F N S F F | 4 5 1 | 1  0  0  0   1  |
Nr. 4: F A F F F D D F N S F A F F F F F F F D D F F A A A F F | 4 5 1 | 1  0  0  0   1  |
Nr. 5: A F F F F F F F A A F F N N S F F F F F F F F F F D D | 2 3 2 | 0  0  0  0   0  |
-----------------------------------------------------------------|-------|-----------------|
Nr. 6: A N N S F F F F D D F F F F F F F F F F A A F F A F F | 2 4 2 | 1  0  0  0   1  |
Nr. 7: F F F N S F F F F F F F N S F F D F D F F F F F A A F F F | 2 2 2 | 0  0  0  0   0  |
Nr. 8: F F F F F F F F D A A F F A A A F N N S F F D F N S D D D | 5 5 3 | 0  0  0  0   0  |
Nr. 9: F F F F D A A A F D D D F F N S D F F A A N N S F F F F | 5 5 3 | 0  0  0  0   0  |
Nr.10: F F A A N N N S F F F F F F D F F F A F F F F F D D A A | 3 5 3 | 2  0  0  0   2  |
-----------------------------------------------------------------|-------|-----------------|
Nr.11: F D D D D F F D F N N S A A F N S A F F F D D F F F A A | 7 5 3 | 1  0  0  0   1  |
Nr.12: F F F F A F F N S F F D D D F A A A A F F F D D F F N N | 5 5 3 | 0  0  0  0   0  |
Nr.13: D D D D A F F F D F A A F F A A F F N N N S F D F F F F | 6 5 3 | 0  0  0  0   0  |
-----------------------------------------------------------------|-------|-----------------|
Samlede pris: 9
```

vp(hol13,hol13t2,14,2,1,step,500,0,2000).

```
Benyttet tid: 506.42
Found a solution with cost 21
Branch-and-bound timeout!

      m t o t f l s m t o t f l s m t o t f l s m t o t f l s | D A N | AV WS FØ EV sum |
-------------------------------------------------------------|-------|-----------------|
Nr. 1: F D D F A F F F F A F F D D N S F F F F F F F A F F D D | 6 3 1 | 2  0  0  0   2  |
Nr. 2: F F D D F A A F F D F F F F F F F A F D D F N S F F F F | 5 3 1 | 1  0  0  0   1  |
Nr. 3: F A F F D A A F D F F F F F D N S F F A A F D F F F F F | 4 5 1 | 1  0  0  0   1  |
Nr. 4: A F F F F F F D D F F F A A F D F F D F F N S F F F A A | 4 5 1 | 1  0  0  0   1  |
Nr. 5: A F F D D F F A A F F F F F F F F F F N N S F F F F F F | 2 3 2 | 0  0  0  0   0  |
-------------------------------------------------------------|-------|-----------------|
Nr. 6: N S F F F F F F F F F D D D F F F F F F F F F A F A A F N | 3 3 2 | 0  1  0  0   1  |
Nr. 7: F F A F F F F F F N S F A A F F F N S F F D F F F F F A A | 1 5 2 | 3  0  0  0   3  |
Nr. 8: F N S F F D D N S A A F F F F F N S A A A F D F D D F F | 5 5 3 | 0  0  0  0   0  |
Nr. 9: F F F N S D D F N S A A F F A A A F F D D A F D N S F F | 5 6 3 | 0  0  0  0   0  |
Nr.10: D D A A F F F F F F N N S F A F F D D F F D A F F N S F | 5 4 3 | 1  0  0  0   1  |
-------------------------------------------------------------|-------|-----------------|
Nr.11: D A F F F N N S F F D A F F D D D D F F F F F A A A N S | 6 5 3 | 0  1  0  0   1  |
Nr.12: F F F A A F F D F D F F N N S F D A A F F F F N S F D D | 5 4 3 | 1  0  0  0   1  |
Nr.13: F F N S N S F A A F D D F F F A A F N S F A F D D D F F | 5 5 3 | 0  0  0  1   1  |
-------------------------------------------------------------|-------|-----------------|
Samlede pris: 13

(Delløsning: pris: 2, tid: 1)
```

D.5 hol13n2

GAMS

```
      m t o t f l s m t o t f l s m t o t f l s m t o t f l s | D A N |
-------------------------------------------------------------|-------|
Nr. 1: F F D A F A A F D F F F F F F D F F D F F D D A F F A A | 6 6 0 |
Nr. 2: A A F F F F F A F F D D D D A F F F A F F F F D D F F F | 6 5 0 |
Nr. 3: S F F A F F F A F F F D A A A F A F F F F F D D F D D D | 6 6 0 |
Nr. 4: S D A F A F F F A F D F D D F F F A F F F F A F F D F F | 5 5 0 |
Nr. 5: A F A F D F F F F F A A F F F F D F D D D F A F D F F F | 6 5 0 |
-------------------------------------------------------------|-------|
Nr. 6: D F D F F D D F D A A F F F D F A F F A A F F F A F F F | 6 6 0 |
Nr. 7: D D F D D D D F F F F A F F F A F F A A A F F F F A F F | 6 6 0 |
Nr. 8: F N N S F F F D F D F F F F F N S D F F F F F A N S A A | 3 3 4 |
Nr. 9: S F F F F F F F F D F F A A F A N S F F F D N S F N N N | 2 3 5 |
Nr.10: S F F F A A A F F N N S F F D F F D N N N S F F A F F F | 2 4 5 |
-------------------------------------------------------------|-------|
Nr.11: N S F N S F F N N S F F F F N S F A F F F A F F F F D D | 2 2 5 |
Nr.12: S F F F N N N S A A F F F F F F D F F D D N S F F A F F | 3 3 4 |
Nr.13: F A F D F F F D F F F N N N S D F N S F F A F N S F F F | 3 2 5 |

Samlede pris: 1
%
% Uden Amin, Dmin
%
%      m t o t f l s m t o t f l s m t o t f l s m t o t f l s | D A N |
%-------------------------------------------------------------|-------|
%Nr. 1: S D F F A F F D F A F D F F F F F F F F F F D F F D D | 6 2 0 |
%Nr. 2: F D D F F F F A F D D F F F F A F D F D D A A A F F F F | 7 5 0 |
%Nr. 3: A F A F F F F F D A A F F F F F D D F F F F F F D F A A | 4 6 0 |
%Nr. 4: S F F A F D D F F F A A F F D D A F F A A F F F D D F F | 6 6 0 |
%Nr. 5: S F F F D F F F F F F A A A F F D A A F F F D F A F D D | 5 6 0 |
%-------------------------------------------------------------|-------|
%Nr. 6: A A F D D D D F A F F F F F F A F A F D D F A F F F F F | 6 6 0 |
%Nr. 7: D F A A F F F F D D F F A A A F F F D F F F D F F A F F | 5 6 0 |
%Nr. 8: S F F N S A A F F F F F F F A F N S D F F D F N N S F F | 2 3 4 |
%Nr. 9: F A F D F F F A F F D N N N N S F N S F F A F F F D F F | 3 3 5 |
%Nr.10: F N N S F F F D F N N S D D F N S F F F F F F F D F A A A | 4 3 5 |
%-------------------------------------------------------------|-------|
%Nr.11: D F F F N N N N N S F D F F F F A F F A A F F F F F F F | 2 3 5 |
%Nr.12: F F D F F A A F A F F F F F D D F F N N N N N S F F F F | 3 3 5 |
%Nr.13: N S F F A F F F F F F F D D F F F F A F F D F A A N N N | 3 4 4 |
%
%Samlede pris: 4
```

vp(hol13,hol13n2,28,0,0,step,1000,0,2000).

```
Benyttet tid: 27.8600000000006
Found a solution with cost 1
Branch-and-bound timeout!

      m t o t f l s m t o t f l s m t o t f l s m t o t f l s | D A N | AV WS FØ EV sum |
-----------------------------------------------------------------|-------|-----------------|
Nr. 1: F F F A F A A F D D D D F F F D F F F A A A F D F F F F | 6 6 0 | 0  0  0  0   0  |
Nr. 2: F F A F F A A F D D D D F F F D F F F A A A F F D F F F | 6 6 0 | 0  0  0  0   0  |
Nr. 3: F F A F F D D F A A A A F F F A F F D D D F F F F D F F | 6 6 0 | 0  0  0  0   0  |
Nr. 4: F A F F F D D F A A A A F F F A F F D D D F F F F D F F | 6 6 0 | 0  0  0  0   0  |
Nr. 5: A F F F F F F D F F F F A A F F D D A F F D D F F F A A | 5 6 0 | 0  0  0  0   0  |
-----------------------------------------------------------------|-------|-----------------|
Nr. 6: F F F F F F F D F F F F A A F F D D A F F D D F F A A A | 5 6 0 | 0  0  0  0   0  |
Nr. 7: A F F F F F F A F F F F D D D F A A F F F F A A F F D D | 5 6 0 | 0  0  0  0   0  |
Nr. 8: D A F F D N N S F F F F F F D F F A F N N S A F F F F F | 3 3 4 | 0  0  0  0   0  |
Nr. 9: F D F D D F F A F F F F N N S F A F F F F F F F F N N | 3 2 4 | 0  0  0  0   0  |
Nr.10: N N N N S F F N S F F F D D A F F F F F F F F F A F D D | 4 2 5 | 1  0  0  0   1  |
-----------------------------------------------------------------|-------|-----------------|
Nr.11: F F D D N S F F N N N S F F N S F F F F F F F A A A F F | 2 3 5 | 0  0  0  0   0  |
Nr.12: D F F F A F F F F F F N S F A N N N S F F F F D D N S F | 3 2 5 | 0  0  0  0   0  |
Nr.13: F D D A A F F F F F F F F F F F F F N S F N N N N S F F | 2 2 5 | 0  0  0  0   0  |
-----------------------------------------------------------------|-------|-----------------|
Samlede pris: 1
```

vp(hol13,hol13n2,14,2,0,step,500,0,2000).

```
Benyttet tid: 19.1499999999996
Found a solution with cost 1
Branch-and-bound timeout!

      m t o t f l s m t o t f l s m t o t f l s m t o t f l s | D A N | AV WS FØ EV sum |
-----------------------------------------------------------------|-------|-----------------|
Nr. 1: F D F F D A A F D F F F F F D D F F F A A A F F A F F F | 5 6 0 | 0  0  0  0   0  |
Nr. 2: F F A F D A A F D F D F F F D D F F F A A A F D F F F F | 6 6 0 | 0  0  0  0   0  |
Nr. 3: F F F F F D D F A A A F F F F A A F F D D F D F D A F F | 6 6 0 | 0  0  0  0   0  |
Nr. 4: F F F A F D D F A A F D F F F A A F F D D F A F D F F F | 6 6 0 | 0  0  0  0   0  |
Nr. 5: F F A F F F F D F D D F A A F F D D D F F F F F F A A A | 6 6 0 | 0  0  0  0   0  |
-----------------------------------------------------------------|-------|-----------------|
Nr. 6: F F F F F F F D F D F D A A A F D D D F F F F F F F A A | 6 5 0 | 0  0  0  0   0  |
Nr. 7: D A F F A F F A F F F F D D A F F A A F F F F F F D D D | 6 6 0 | 0  0  0  0   0  |
Nr. 8: F F D D A N N S F F F A F F F F F F F N N S F D F F F F | 3 2 4 | 0  0  0  0   0  |
Nr. 9: A F D D F F F A F F F F N N S F F F F F F F A F F N N | 2 3 4 | 0  0  0  0   0  |
Nr.10: N N S A F F F F F F F F D D N N N S F F F F A A F F D D | 4 3 5 | 1  0  0  0   1  |
-----------------------------------------------------------------|-------|-----------------|
Nr.11: D D N N S F F F F F A A F F F F F F N N S F N S F A F F F | 2 3 5 | 0  0  0  0   0  |
Nr.12: F A F F N S F N N S F F F F F F F A F F D N N S D F F | 2 2 5 | 0  0  0  0   0  |
Nr.13: A F F F F F F F F N N N S F F F F A F F F D D F N N S F | 2 2 5 | 0  0  0  0   0  |
-----------------------------------------------------------------|-------|-----------------|
Samlede pris: 1

(Delløsning: pris: 0, tid: 7)
```

vp(hol13,hol13n2,28,0,1,step,1000,0,2000).

```
Benyttet tid: 39.7200000000012
Found a solution with cost 5
Branch-and-bound timeout!

      m t o t f l s m t o t f l s m t o t f l s m t o t f l s | D A N | AV WS FØ EV sum |
-----------------------------------------------------------------|-------|-----------------|
Nr. 1: F F F D D A A F F F D D F F D F F F D A A A F D F F F F | 7 5 0 | 1  0  0  0   1  |
Nr. 2: F F A F D F F F A A F F D D F A A A F F F F F F D D D D | 7 6 0 | 1  0  0  0   1  |
Nr. 3: F F F A F F F F A A F F D D F A A A F F F F F F D D D D | 6 6 0 | 0  0  0  0   0  |
Nr. 4: F A F F F A A F D D F F F F D D D F F A A A F F F F F F | 5 6 0 | 0  0  0  0   0  |
Nr. 5: F D D D A F F F F F F F F A A F F F F F F F D D D F F A A | 6 5 0 | 0  0  0  0   0  |
-----------------------------------------------------------------|-------|-----------------|
Nr. 6: F F A F F D D A F D A A F F A F F F F D D F F F F A F F | 5 6 0 | 0  0  0  0   0  |
Nr. 7: D D D A A F F D F F D F F F F F F F F F F D A A A F F F | 6 5 0 | 0  0  0  0   0  |
Nr. 8: N N S F F D D N N S F A F F F F F F F F D D F F F F A F F | 4 2 4 | 1  0  0  0   1  |
Nr. 9: F A F F F N N S F F F D F F A F F F D N N S A F F F F F | 2 3 4 | 0  0  0  0   0  |
Nr.10: F F F F F F F D D F F F A A F N N N S F F N N S F F A A | 2 4 5 | 1  0  0  0   1  |
-----------------------------------------------------------------|-------|-----------------|
Nr.11: D F N N N S F F F N S F F F F F F F N S F F D A A F F F | 2 2 5 | 0  0  0  0   0  |
Nr.12: A F F F F F F A F F N S N N S D F D A F F F F F F F N N | 2 3 5 | 0  0  0  1   1  |
Nr.13: A F F F F F F F F F A N S F N S D D A F F F F N N N S F | 2 3 5 | 0  0  0  0   0  |
-----------------------------------------------------------------|-------|-----------------|
Samlede pris: 5
```

vp(hol13,hol13n2,14,2,1,step,1000,0,2000).

```
Benyttet tid: 66.5599999999977
Found a solution with cost 1
Branch-and-bound timeout!

      m t o t f l s m t o t f l s m t o t f l s m t o t f l s | D A N | AV WS FØ EV sum |
-------------------------------------------------------------|-------|-----------------|
Nr. 1: F F F D D F F A A F D A F F F F F F F D D F A A F A F F | 5 6 0 | 0  0  0  0   0  |
Nr. 2: F F A F F D D D F F A A F F F F D D D A A F F F F F F F | 6 5 0 | 0  0  0  0   0  |
Nr. 3: F F D A F D D F F A A F F F D F F A A F F F F F A F D D | 6 6 0 | 0  0  0  0   0  |
Nr. 4: F A F F F F F A A F F D D D F A A F D F F D D F F A F F | 6 6 0 | 0  0  0  0   0  |
Nr. 5: F F A F F F F D D F D F A A A F F F F F F D D D F F A A | 6 6 0 | 0  0  0  0   0  |
-------------------------------------------------------------|-------|-----------------|
Nr. 6: F D F A A F F F F F F D D F A A A A F F F F F D F D D  | 6 6 0 | 0  0  0  0   0  |
Nr. 7: F F D D F A A F F D F F F F F D D F F A A A F F F D F F | 6 5 0 | 0  0  0  0   0  |
Nr. 8: D F F F D F F N N S F F N N S F F F F F F F A A A F F F | 2 3 4 | 0  0  0  0   0  |
Nr. 9: N N S F F A A F F F F D F F F F N N S D D F F F F F F F | 3 2 4 | 0  0  0  0   0  |
Nr.10: F D F F A F F F F D F F A A A F F F F F F F F N N N N N | 2 4 5 | 1  0  0  0   1  |
-------------------------------------------------------------|-------|-----------------|
Nr.11: D A N N S F F F D A F F F F F F F N N N S F F F D F F  | 3 2 5 | 0  0  0  0   0  |
Nr.12: A F F F N N N S F F F F F F N N S D F F F A F D D F F F | 3 2 5 | 0  0  0  0   0  |
Nr.13: A F F F F F F F F N N N S F D D F F F F F N N S F F A A | 2 3 5 | 0  0  0  0   0  |
-------------------------------------------------------------|-------|-----------------|
Samlede pris: 1
```

D.6 hol13f

GAMS

```
Resource limit ecxeeded, no integer solution exists
%
%
% Uden Amin, Dmin
%
%      m t o t f l s m t o t f l s m t o t f l s m t o t f l s | D A N |
%-------------------------------------------------------------|-------|
%Nr. 1: F D D F F F F F F F D F N N S A F F F F F A F F F F D D | 5 2 2 |
%Nr. 2: F N S F F D D F A F F F F F F F F F D D N S A A F F F | 4 3 2 |
%Nr. 3: S A A F D F F A F D F F A A F N S F D F F D N S F F D F | 5 5 2 |
%Nr. 4: F D D F F F F N N S F A F F F D F A A A A F F F F F F F | 3 5 2 |
%Nr. 5: A F F D F N N S D D A F F F F F F A F F D D F F F A A F F | 5 5 2 |
%-------------------------------------------------------------|-------|
%Nr. 6: F F F F F D D D F A A A F F D F A N S F F F F F F D F N | 5 4 2 |
%Nr. 7: A F F A F F F F F F N S F A A A F D D F F F F D N S F F D | 4 5 2 |
%Nr. 8: S A F F D F F F D F N N S F F F F D F F F D F D F A A A | 5 4 2 |
%Nr. 9: N S F F A A A F F F D D F F F D N S F F A F F A F D F F | 4 5 2 |
%Nr.10: F F F A A A F D F F F F F F D F F A N N N S A F D F F F | 3 5 3 |
%-------------------------------------------------------------|-------|
%Nr.11: D F N S F F F F A F F F D D N S F F A F F F A F D F A A | 4 5 2 |
%Nr.12: S F A N S F F A F F F F D D F A F F D F F A F D F N N S | 4 4 3 |
%Nr.13: D F F D N S A F F A F D F F A F D F F A F F D F N S F F | 5 4 2 |
%
%Samlede pris: 11
```

vp(hol13,hol13f,28,0,0,step,1000,0,2000).

```
Benyttet tid: 39.9199999999983
Found a solution with cost 11
Branch-and-bound timeout!

      m t o t f l s m t o t f l s m t o t f l s m t o t f l s | D A N | AV WS FØ EV sum |
-------------------------------------------------------------|-------|-----------------|
Nr. 1: F F F D F N N S D D D F F F F D F F F A A A A A F F F F | 5 5 2 | 0  0  0  0   0  |
Nr. 2: F F F F D F F D A A F F N N S D D D F F F F F F A F A A | 5 5 2 | 0  0  0  0   0  |
Nr. 3: A F F F F F F F D A A F D F F D A A F F N N S D D F F F F | 5 5 2 | 0  0  0  0   0  |
Nr. 4: F D A F F F F F F D D F F A A F A A F D F F D F F F F N N | 5 5 2 | 0  0  0  0   0  |
Nr. 5: N N S F F A A F F F F D D F F F F F D D F A A F F F F D A F F | 5 5 2 | 0  0  0  0   0  |
-------------------------------------------------------------|-------|-----------------|
Nr. 6: D F N N S A A F F F A A F F D F F F F D D D F F F F F F F | 5 4 2 | 0  0  0  0   0  |
Nr. 7: F F F D F D D N N S A F F F F F F A F F F F D D F F A A | 5 4 2 | 0  0  0  0   0  |
Nr. 8: F F D F F F D D A F N N S F F A F F A F D D F A A F F F F | 5 5 2 | 0  0  0  0   0  |
Nr. 9: F D F F F F F F F F F A A A N N S F A F F F F F D D D D | 5 4 2 | 0  0  0  0   0  |
Nr.10: F F F F A F F A F F F F F D D A F N N S F F A F F F D D D | 5 4 2 | 0  0  0  0   0  |
-------------------------------------------------------------|-------|-----------------|
Nr.11: D A F A A F F F F F F F D D F F F F A F F N S F N S F F | 3 4 2 | 1  0  0  0   1  |
Nr.12: F F D A N S F F F F F F F F F F F F F N S F D N S A A F F | 2 3 3 | 1  0  0  0   1  |
Nr.13: A A A F D F F F F F F N S F F F F F F F F F N S N S F | 1 3 3 | 2  0  0  1   3  |
-------------------------------------------------------------|-------|-----------------|
Samlede pris: 5
```

vp(hol13,hol13f,14,2,0,step,500,0,2000).

```
Benyttet tid: 507.310000000001
Found a solution with cost 2
Branch-and-bound timeout!

      m t o t f l s m t o t f l s m t o t f l s m t o t f l s | D A N | AV WS FØ EV sum |
-------------------------------------------------------------|-------|-----------------|
Nr. 1: N S F F D A A F D F F F F F N S F F D A A F D D F F F F | 5 4 2 | 0  0  0  0   0  |
Nr. 2: F N S F F A A F D D F A F F F N S F F A A F D D F F F F | 4 5 2 | 0  0  0  0   0  |
Nr. 3: A F N S D F F D F F F F A A F D D F F F F F N S F F D A A | 5 5 2 | 0  0  0  0   0  |
Nr. 4: A F D F F F F N S D F F A A F F N S D F F D F F F D A A | 5 5 2 | 0  0  0  0   0  |
Nr. 5: F F F N S D D F F A A F F F A F F N S D D D F A A F F F | 5 5 2 | 0  0  0  0   0  |
-------------------------------------------------------------|-------|-----------------|
Nr. 6: F F F A F D D A N S F F F F A A A F F D D F N S F F F F | 4 5 2 | 0  0  0  0   0  |
Nr. 7: F F F A A F F F F N S D D D F A A F F F F F F N S F D D | 5 4 2 | 0  0  0  0   0  |
Nr. 8: F F F F A F F F F A N S D D F F F F A F F F F A N S D D | 4 4 2 | 0  0  0  0   0  |
Nr. 9: F F D D N S F A A F F F F F D D D A A F F F F F F N S F | 5 4 2 | 0  0  0  0   0  |
Nr.10: D D A F F F F D A F F N S F F F F A F F F A F F D F N S | 4 4 2 | 0  1  0  0   1  |
-------------------------------------------------------------|-------|-----------------|
Nr.11: D D F D F F F F F F A A N N S F F D F F F F A F A A F F | 4 5 2 | 0  0  0  0   0  |
Nr.12: F A A F F F F F F F D D F F F F F F N N N S A F D F F F | 3 3 3 | 0  0  0  0   0  |
Nr.13: F A F F F N N S F F D F F F D F F D F F F A F F F A F N | 3 3 3 | 0  1  0  0   1  |
-------------------------------------------------------------|-------|-----------------|
Samlede pris: 2

(Delløsning: pris: 2, tid: 7)
```

vp(hol13,hol13f,28,0,1,step,1000,0,2000).

```
Benyttet tid: 2.47000000000116
Found a solution with cost 18
Branch-and-bound timeout!

      m t o t f l s m t o t f l s m t o t f l s m t o t f l s | D A N | AV WS FØ EV sum |
-------------------------------------------------------------|-------|-----------------|
Nr. 1: F F D D D A A F F F F F F F F F F A A F D N N S F F F | 4 4 2 | 0  0  0  0   0  |
Nr. 2: D D F A F F F F N S F F A A N S A A F F F F F D D D D D | 7 5 2 | 2  0  0  0   2  |
Nr. 3: A N N S A F F F F F F F D D F F F F F F F F F D F A F F | 3 3 2 | 2  0  0  0   2  |
Nr. 4: F A A F F F F F F F F F D N N S D F F A F F D F F F D A A | 4 5 2 | 0  0  0  0   0  |
Nr. 5: A A A A F F F D D D D A F F F A A A F F F F F F F F N N | 4 8 2 | 3  0  0  0   3  |
-------------------------------------------------------------|-------|-----------------|
Nr. 6: F F F F F N N S A A A F F F D A F F D D D F F F D F F F | 5 4 2 | 0  0  0  0   0  |
Nr. 7: D D D D A F F A F F F N S F A F D D F F F F A F F N S F | 6 4 2 | 1  0  0  0   1  |
Nr. 8: F F F F D D D F A A A A F F F F F F F D D N N S F F F F | 5 4 2 | 0  0  0  0   0  |
Nr. 9: F F F F F F F D F F N S A A F N S D F F F D D F F F A A | 4 4 2 | 0  0  0  0   0  |
Nr.10: F F F F F A A F D D D D F F F F N N S A A F F F F F F F | 4 4 2 | 0  0  0  0   0  |
-------------------------------------------------------------|-------|-----------------|
Nr.11: N S F N S F F F F F F F D D A F F F A F F A A F F F D D | 4 4 2 | 0  0  0  0   0  |
Nr.12: F F F F N S F A F N S F F F D D D F N S F A F A A F F F | 3 4 3 | 0  0  0  0   0  |
Nr.13: F F F F F D D N S F F F F F F F F F D N N S F A A A F F | 3 3 3 | 0  0  0  0   0  |
-------------------------------------------------------------|-------|-----------------|
Samlede pris: 8
```

vp(hol13,hol13f,14,2,1,step,500,0,2000).

```
Benyttet tid: 610.800000000003
Found a solution with cost 2
Branch-and-bound timeout!

      m t o t f l s m t o t f l s m t o t f l s m t o t f l s | D A N | AV WS FØ EV sum |
-------------------------------------------------------------|-------|-----------------|
Nr. 1: N S A F F D D F F F F A F F F A A F N S F A F D D F F F | 4 5 2 | 0  0  0  0   0  |
Nr. 2: F A F A N S F A F D D D F F D F F F F D D A N S F F F F | 6 4 2 | 1  0  0  0   1  |
Nr. 3: A F F D D F F F A A F D F F A F F F A N N S D F F F F F | 4 5 2 | 0  0  0  0   0  |
Nr. 4: F D D F F F F D A A F F D D F F A F F F F F F F F A N N | 5 4 2 | 0  0  0  0   0  |
Nr. 5: A F F N S F F D D F F F A A F D D D F F F D F N S F A A | 6 5 2 | 1  0  0  0   1  |
-------------------------------------------------------------|-------|-----------------|
Nr. 6: F F D D F A A F N S F F F F F F D A F F F D F F A N S F | 4 4 2 | 0  0  0  0   0  |
Nr. 7: D A A F F F F F F F F D N S F F N S A F F F F A F D D F F | 4 4 2 | 0  0  0  0   0  |
Nr. 8: F F F A A F F N S F F F D D F F N S A F F F A A F F D D | 4 5 2 | 0  0  0  0   0  |
Nr. 9: F N S F F D D F F F A A F F F F F F F D D N S F A A F F | 4 4 2 | 0  0  0  0   0  |
Nr.10: F F F F D A A F D N S F F F N S F F D A A F D D F F F F | 5 4 2 | 0  0  0  0   0  |
-------------------------------------------------------------|-------|-----------------|
Nr.11: D D F F F F F F F F N S A A A F F D D F F F F F N S A A | 4 5 2 | 0  0  0  0   0  |
Nr.12: F F N S F F F A F F A F N N S A F F F F F F F A F D D D | 3 4 3 | 0  0  0  0   0  |
Nr.13: F F F F A N N S F D F F F F D D F N S A A F F F F F F F | 3 3 3 | 0  0  0  0   0  |
-------------------------------------------------------------|-------|-----------------|
Samlede pris: 2

(Delløsning: pris: 2, tid: 2)
```

D.7 hol13fa

GAMS

```
      m t o t f l s m t o t f l s m t o t f l s m t o t f l s | D A N |
-------------------------------------------------------------|---------|
Nr. 1: F D F F F F F F F F F N N N S A F F D F F D A A A F D D | 5 4 3 |
Nr. 2: F F F D D F F A F D D A A A F F F F A F F F N S F N S D | 5 5 2 |
Nr. 3: S A F N S F F A F D F A F F A F D F F F N S D D F F F F | 4 4 2 |
Nr. 4: F F D F F F F N N S F D F F D F A F A A A F F F F D F F | 4 4 2 |
Nr. 5: D F N S F D N S F A A F F F A F F D F D D F A F F A F F | 5 5 2 |
-------------------------------------------------------------|-------|
Nr. 6: F F D F N S D D F F A F F F F F A F F F F F D F A F A N | 4 4 2 |
Nr. 7: A A F F F F F F F D F D F D D F A F A F F F F F N N S D A | 5 5 2 |
Nr. 8: D D A F A A A F A F F F F F D F N S F F F N S F D F F F | 4 5 2 |
Nr. 9: A F F A F D D D F N S F F F F F F A N N S D F F F A F F | 4 4 3 |
Nr.10: F F A F F N S F F F N S F F F D F D F A A A F D D F F F | 4 4 2 |
-------------------------------------------------------------|-------|
Nr.11: F F F A F F F F D F F D A A F D F N S F F A F F F D N S | 4 4 2 |
Nr.12: N S F D D F F F A F F F D D F N S F F F F F F A F F A A | 4 4 2 |
Nr.13: F N S F A A A F F A F F F F N S D F D D D F F F F F F F | 4 4 2 |

Samlede pris: 14
%
% Uden Amin, Dmin
%
%      m t o t f l s m t o t f l s m t o t f l s m t o t f l s | D A N |
%-------------------------------------------------------------|-------|
%Nr. 1: F N S D F A A A F A N S F F A F F F F D D F F D F F F F | 4 5 2 |
%Nr. 2: D F A F D F F F A F A F A A F F N N S F F F F D A F F F | 3 6 2 |
%Nr. 3: A F F N S F F F A F F N S F F D D F D F F F F A F F A A | 3 5 2 |
%Nr. 4: A F N S F F F F D D D F F F D F F A N S D A F F F A F F | 5 4 2 |
%Nr. 5: F F F F F N S D F F F A F F F F F D F F A N S D A F D F F | 4 3 2 |
%-------------------------------------------------------------|-------|
%Nr. 6: F F D F F F F N S F D D A A A F A F A F F D F F D N S F | 5 5 2 |
%Nr. 7: F A F A F A A F D F F D F F F N S F F D A F N S D D F F | 5 5 2 |
%Nr. 8: F D F F D D F A F D F F F F F F F A F F F F F A N N S A A | 4 5 2 |
%Nr. 9: F D F F A F F D N S F F N N S A F A F F F D A F F F D D | 5 4 3 |
%Nr.10: F A F F F F F F F A F F D D F D F D F F F F F F F F N N | 4 2 2 |
%-------------------------------------------------------------|-------|
%Nr.11: N S F D F F F F F F F F D D N S F F A F F A F F A F D D | 5 3 2 |
%Nr.12: S F D F F D D D F N S A F F D F F F D A A N S F F A F F | 6 4 2 |
%Nr.13: D F A A A N N S F F F A F F F A F D F N S F D F F F F F | 3 5 3 |
%
%Samlede pris: 15
```

vp(hol13,hol13fa,28,0,0,step,1000,0,2000).

```
no
```

vp(hol13,hol13fa,14,2,0,step,500,0,2000).

```
Benyttet tid: 504.740000000002
Found a solution with cost 2
Branch-and-bound timeout!

      m t o t f l s m t o t f l s m t o t f l s m t o t f l s | D A N | AV WS FØ EV sum |
-------------------------------------------------------------|-------|-----------------|
Nr. 1: N S F F D A A F D F F F F F N S F F D A A F F D F F F F | 4 4 2 | 0  0  0  0   0  |
Nr. 2: F F N S F F F D D F F D A A A N S F D F F F D F F F A A | 5 5 2 | 0  0  0  0   0  |
Nr. 3: A N S F D F F D F F F F A A F D D F F F F D N S F F A A | 5 5 2 | 0  0  0  0   0  |
Nr. 4: F D A F F F F N S A F F D D F A N S A F F F F F F F D D | 5 4 2 | 0  0  0  0   0  |
Nr. 5: F F D N S A A F F D D F F F D D F N S A A F F F A F F F | 5 5 2 | 0  0  0  0   0  |
-------------------------------------------------------------|-------|-----------------|
Nr. 6: F F F F F D D A A N S F F F F F A F A F F D F N S F D D | 5 4 2 | 0  0  0  0   0  |
Nr. 7: F F F F F F D D F A A N S F F D A A F F F F F D D F N S F | 5 4 2 | 0  0  0  0   0  |
Nr. 8: F F F F A F F A N S F F D D F F D D F F F F F A A A N S | 4 5 2 | 0  1  0  0   1  |
Nr. 9: F F D A N S F F F D A A F F A F F F F D D F F A N S F F | 4 5 2 | 0  0  0  0   0  |
Nr.10: D D A F F F F F F F A N S F F F F A F F F A A F D D F N | 4 5 2 | 0  1  0  0   1  |
-------------------------------------------------------------|-------|-----------------|
Nr.11: F F F A A F F F F F D D N N S F F F F F F A A F D D F F | 4 4 2 | 0  0  0  0   0  |
Nr.12: D A F D F F F F F F F A F F F F F D N N N S F F F A F F | 3 3 3 | 0  0  0  0   0  |
Nr.13: A A F D F N N S F F F F F F F F F A F D D N S F F F F F | 3 3 3 | 0  0  0  0   0  |
-------------------------------------------------------------|-------|-----------------|
Samlede pris: 2

(Delløsning: pris: 2, tid: 1)
```

vp(hol13,hol13fa,28,0,1,step,1000,0,2000).

```
Benyttet tid: 210.07
Found a solution with cost 4
Branch-and-bound timeout!

      m t o t f l s m t o t f l s m t o t f l s m t o t f l s | D A N | AV WS FØ EV sum |
-----------------------------------------------------------------|-------|-----------------|
Nr. 1: A F F N S D D N S F F F F F F F F F F D D F F A A A F F | 4 4 2 | 0  0  0  0   0  |
Nr. 2: F F F F A F F F D F D F A A F A A F D F F F D D F F N N | 5 5 2 | 0  0  0  0   0  |
Nr. 3: F F D A F F F D A A F F D D F N S F D F F D N S F F A A | 6 5 2 | 1  0  0  0   1  |
Nr. 4: D A A F F F F F F F F F D F F F F N N S D D A F F F A F F | 4 4 2 | 0  0  0  0   0  |
Nr. 5: F F D D F F F F F F F F N N S F A A A F F D A F D D F F | 5 4 2 | 0  0  0  0   0  |
-----------------------------------------------------------------|-------|-----------------|
Nr. 6: F F A A F D D F N N S F F F A F F F F F F F A A F F D D | 4 5 2 | 0  0  0  0   0  |
Nr. 7: F D F F D F F D A A F N S F A A F D F F F F D F F N S F | 5 4 2 | 0  0  0  0   0  |
Nr. 8: A A F F A A A F F F F F F F F F F F N N S F D D D F F | 3 5 2 | 1  0  0  0   1  |
Nr. 9: F D F F F N N S D F A A F F D F D D F A A A F F A F F F | 5 6 2 | 1  0  0  0   1  |
Nr.10: N N S F F F F F F D D F A A F D D F F F F F F F F F A A | 4 4 2 | 0  0  0  0   0  |
-----------------------------------------------------------------|-------|-----------------|
Nr.11: D F F D D A A F F F F D F F F F F F F A A N S F N S F F | 4 4 2 | 0  0  0  0   0  |
Nr.12: F F F F F F F A F F N S D D N S F A A F F F F N S F D D | 4 3 3 | 0  0  0  0   0  |
Nr.13: F F N S N S F A F D A A F F D D F F N S F F F F F F F F | 3 3 3 | 0  0  0  1   1  |
-----------------------------------------------------------------|-------|-----------------|
Samlede pris: 4
```

vp(hol13,hol13fa,14,2,1,step,500,0,2000).

```
Benyttet tid: 701.380000000001
Found a solution with cost 4
Branch-and-bound timeout!

      m t o t f l s m t o t f l s m t o t f l s m t o t f l s | D A N | AV WS FØ EV sum |
-----------------------------------------------------------------|-------|-----------------|
Nr. 1: F F F N S F F A F A F F A A F F F N S F F F F D D D D D | 5 4 2 | 0  0  0  0   0  |
Nr. 2: A F N S F F F D D F F D D D F A F A A F F F A F F N S F | 5 5 2 | 0  0  0  0   0  |
Nr. 3: D D F F A F F F F F D F A N N S F D F D F F A F F F D A A | 6 5 2 | 1  0  0  0   1  |
Nr. 4: F F D F F F F A A F D F F F D F F D A N N S F F F A F F | 4 4 2 | 0  0  0  0   0  |
Nr. 5: A F F D N S F F F F A F F F F F A A F D D F D N S F F F | 4 4 2 | 0  0  0  0   0  |
-----------------------------------------------------------------|-------|-----------------|
Nr. 6: F A A A F F F D F D F N S F F F N S D F F D F D A F A A | 5 6 2 | 1  0  0  0   1  |
Nr. 7: F F A A F D D F N S F D F F F D F F F F F F F F A A N S | 4 4 2 | 0  1  0  0   1  |
Nr. 8: N S F F A F F F F F F F A A A F D D F F F F F F A N S D D | 4 5 2 | 0  0  0  0   0  |
Nr. 9: F N S F F D D F F A A A F F N S A F F D D F F F F F F F | 4 4 2 | 0  0  0  0   0  |
Nr.10: D D D F F F F F F F N S F D D A F F F F F F F A A A F F F N | 5 4 2 | 0  1  0  0   1  |
-----------------------------------------------------------------|-------|-----------------|
Nr.11: F A F D F A A N S F D F F F F A F F N S F D D F F F F F | 4 4 2 | 0  0  0  0   0  |
Nr.12: F F F F D A A F D F N S F F F N S F F A A F N S D F F F | 3 4 3 | 0  0  0  0   0  |
Nr.13: F F F F D N N S A F F F F F D D F F F A A N S F F F F F | 3 3 3 | 0  0  0  0   0  |
-----------------------------------------------------------------|-------|-----------------|
Samlede pris: 4

(Delløsning: Pris: 4, tid: 3)
```

D.8 hol13f1

GAMS

```
No integer feasible solution exists.
Resource limit exceeded, no integer solution found
```

vp(hol13,hol13f1,28,0,0,step,1000,0,2000).

```
Benyttet tid: 321.44
Found a solution with cost 20
Branch-and-bound timeout!

      m t o t f l s m t o t f l s m t o t f l s m t o t f l s | D A N | AV WS FØ EV sum |
-----------------------------------------------------------------|-------|-----------------|
Nr. 1: F F F D D N N S F D D D F F F D F F F A A A A F A F F F | 6 5 2 | 1  0  0  0   1  |
Nr. 2: A F F F F F F D F A A F N N S D D D F F F F F D F F A A | 5 5 2 | 0  0  0  0   0  |
Nr. 3: A F D F F A A F F D D D F F D A A F F N N S F F F F F F | 5 5 2 | 0  0  0  0   0  |
Nr. 4: F F F F F A A F F A A F F F D F D D D F F F F F D A N N | 5 5 2 | 0  0  0  0   0  |
Nr. 5: N N S F F D D F F F F F F F F A A F F A A F D D F D F F | 5 4 2 | 0  0  0  0   0  |
-----------------------------------------------------------------|-------|-----------------|
Nr. 6: F F N N S D D A F F F A F F A F F A F D D F F F F A F F | 4 5 2 | 0  0  0  0   0  |
Nr. 7: F A F F F F F D F N N S A A F F F F D F F D D F F F A A | 4 5 2 | 0  0  0  0   0  |
Nr. 8: F D F F F F F F F F F A A A N N S F A F F F F F D D D D | 5 4 2 | 0  0  0  0   0  |
Nr. 9: F F F F F F F N N S F F D D F F F A A F F F A A F F D D | 4 4 2 | 0  0  1  0   1  |
Nr.10: F F F D D F F A A F F F D D A F N N S F F A F F F F F F | 4 4 2 | 0  0  1  0   1  |
-----------------------------------------------------------------|-------|-----------------|
Nr.11: F F D A N S F F A F F F F F F F F F F D D N S A A F F F | 3 4 2 | 1  0  1  0   2  |
Nr.12: D D A A A F F F D F F N S F F F F F F F F D F N S N S F | 4 3 3 | 0  0  1  1   2  |
Nr.13: D A A F A F F F D F F F F F F F F N S F F N S N S F F | 2 3 3 | 1  0  1  1   3  |
-----------------------------------------------------------------|-------|-----------------|
Samlede pris: 10
```

vp(hol13,hol13f1,14,2,0,step,500,0,2000).

```
no
```

vp(hol13,hol13f1,28,0,1,step,1000,0,2000).

```
Benyttet tid: 9.61000000000013
Found a solution with cost 11
Branch-and-bound timeout!

      m t o t f l s m t o t f l s m t o t f l s m t o t f l s | D A N | AV WS FØ EV sum |
-------------------------------------------------------------|-------|-----------------|
Nr. 1: F F F A A F F N N S F F D D F F F A A F F A F F F F D D | 4 5 2 | 0  0  1  0   1  |
Nr. 2: N N S F F F F F F F D D D D F A A A A F F F F F F D F F | 5 4 2 | 0  0  0  0   0  |
Nr. 3: D F N N S D D F A F F F F F F F F F D D A F F F A F F | 5 3 2 | 1  0  1  0   2  |
Nr. 4: F A A F F N N S D F F F F F F F D F D D D F A A A A F F | 5 6 2 | 1  0  1  0   2  |
Nr. 5: F F A F F F F F F D D D A A F D F F F F F F F N N S A A | 4 5 2 | 0  0  0  0   0  |
-------------------------------------------------------------|-------|-----------------|
Nr. 6: F F D D F A A F F F F F F F F N S F F A A N S D D F F F | 4 4 2 | 0  0  0  0   0  |
Nr. 7: F A F F F A A F F N N S F F D D D F F A A F D F F D F F | 5 5 2 | 0  0  0  0   0  |
Nr. 8: A F F F D D D F D A A A F F A F F F F F F F F F F N N | 4 5 2 | 0  0  1  0   1  |
Nr. 9: A F F F N S F D F A A A F F D A F F N S F D D F F F F F | 4 5 2 | 0  0  0  0   0  |
Nr.10: F F F F A F F A F F F F N N S F F D D F F F F D D F A A | 4 4 2 | 0  0  0  0   0  |
-------------------------------------------------------------|-------|-----------------|
Nr.11: F D D D D F F D F F F F F F F A F F N N S A A A F F F | 5 4 2 | 0  0  0  0   0  |
Nr.12: F F F F F F F F F D F F A A A F N N S F F F N S F F D D | 3 3 3 | 0  0  0  0   0  |
Nr.13: D D F A F F F A A F F N S F N S F D F F F D F F F N S F | 4 3 3 | 0  0  1  0   1  |
-------------------------------------------------------------|-------|-----------------|
Samlede pris: 7
```

vp(hol13,hol13f1,14,2,1,step,500,0,2000).

```
no
```

D.9 hol130

GAMS

```
Iteration limit exceeded
```

vp(hol130,hol130,28,0,0,step,1000,0,2000).

```
*** Overflow of the global/trail stack in spite of garbage collection!
```

vp(hol130,hol130,14,2,0,step,500,0,2000).

```
no
(Delløsning: pris: 20, tid: 22)
```

vp(hol130,hol130,28,0,1,step,1000,0,2000).

```
*** Overflow of the global/trail stack in spite of garbage collection!
```

vp(hol130,hol130,14,2,1,step,500,0,2000).

```
no
```

D.10 var13b

GAMS

```
      m t o t f l s m t o t f l s m t o t f l s | D A N |
-----------------------------------------------------------|-------|
Nr. 1: N S F F F F F F N S A F F D A F F F D D F F D A F F F | 4 3 2 |
Nr. 2: F D A N N S F D D F A F F F F F F A F D D F F A F F F F | 5 4 2 |
Nr. 3: S F D D F F F D D F F F A F F F A F F F F A F F F N N | 4 3 2 |
Nr. 4: F F A F F F F A F F D F D A F F D N S F F F N S D F A F | 4 4 2 |
Nr. 5: F N N S F F F D F F F F D F A F F F A F F F F D D D A A | 5 4 2 |
-----------------------------------------------------------|-------|
Nr. 6: D F F D A F F D D F A A F F N S D F N S F F F A F F F F | 5 4 2 |
Nr. 7: F D D F F F F F F F A F F A A F F N S F F F D F N S D F D | 5 3 2 |
Nr. 8: S F F D N S F N S F A A F F A F F D D D A F F F F F F F | 4 4 2 |
Nr. 9: F F D A F N S F D D F F F F D F F A F A A F D F N S F F | 5 4 2 |
Nr.10: F D F A A N S F D F F D F F F F A N S F F A F F A F D D | 5 5 2 |
-----------------------------------------------------------|-------|
Nr.11: D F A A F D F F F F A F F F N N S F A F F D F D F D F F | 5 4 2 |
Nr.12: A A F F A A F A F F N S F F F D F D F F N S D F F D F F | 4 5 2 |
Nr.13: A A F N S N S F F F D D F F F F F F F A A F F F D F F F F | 3 4 2 |

Samlede pris: 20
```

vp(var13b,hol13,28,0,0,step,1000,0,2000).

```
no
```

vp(var13b,hol13,14,2,0,step,500,0,2000).

```
no
```

vp(var13b,hol13,28,0,1,step,1000,0,2000).

```
no
```

vp(var13b,hol13,14,2,1,step,500,0,2000).

```
no
```

D.11 sim10

GAMS

```
      m t o t f l s m t o t f l s m t o t f l s m t o t f l s | D A N |
-----------------------------------------------------------------|-------|
Nr. 1: A N S F D F F F F N S F F F F F F F A F F N S F A F D D | 3 3 3 |
Nr. 2: S A F F F F F F F F D N N N S F D D F F F A F F F F F F | 3 2 3 |
Nr. 3: F F F A F A A F N S F F F F D F F F F D D F N S F F F F | 3 3 2 |
Nr. 4: F F A F F D D F D F N S F F F F F F F N N S A F F F F F | 3 2 3 |
Nr. 5: D F F F N S F D A F F D F F F N S F F F F F F F N S F A A | 3 3 3 |
-----------------------------------------------------------------|-------|
Nr. 6: S D F F A F F F F F A F F F F N S A F N S F F D F N S F F | 2 3 3 |
Nr. 7: F F N S F F F F F F D A F F F F D N N S A A F F F F D F F | 3 3 3 |
Nr. 8: S F F D F N N N S F F A F F A A F F F F F F F D F F F F | 2 3 3 |
Nr. 9: F F D F F F F F A F F F F A A F F F F D F F F F F D N N N | 3 3 3 |
Nr.10: N S F N S F F F F F F F D D F F F A F F F D F A F A F F | 3 3 2 |
-----------------------------------------------------------------|-------|

Samlede pris: 2
```

vp(sim10,sim10,28,0,0,step,1000,0,2000).

```
Benyttet tid: 9.85999999999967
Found a solution with cost 0
      m t o t f l s m t o t f l s m t o t f l s | D A N | AV WS FØ EV sum |
-----------------------------------------------------------------|-------|-----------------|
Nr. 1: F F A F F N N S D D F F F F F F D A A A F F F F F F | 4 4 2 | 0  0  0  0   0  |
Nr. 2: F A F F F F F F F A F D N N S D F F F F F D D F F F A A | 4 4 2 | 0  0  0  0   0  |
Nr. 3: N S F F F A A F F F F F F F F D D F N N S F D F F F F | 3 2 3 | 0  0  0  0   0  |
Nr. 4: D N S F F D D F A F A F F F F F F F F F F F F A N N | 3 3 3 | 0  0  0  0   0  |
Nr. 5: F F N N S F F N S F F F A A F F F F F F F F F D D D | 3 2 3 | 0  0  0  0   0  |
-----------------------------------------------------------------|-------|-----------------|
Nr. 6: F F F F F F F A N S F F D D N N S A F F F F F F D F F F | 3 2 3 | 0  0  0  0   0  |
Nr. 7: A F F F F N S F F F N N S F F F A A F F D D F F F F F F F | 2 3 3 | 0  0  0  0   0  |
Nr. 8: F F F F D F F D F F F N S F A F N N S F F F A A F F F F | 2 3 3 | 0  0  0  0   0  |
Nr. 9: F F F D A F F F F F F F A F F D F F F N S F N N S A F F F | 2 3 3 | 0  0  0  0   0  |
Nr.10: F D D A F F F F F F F F F F F F F A F F F F N N N S F | 2 2 3 | 0  0  0  0   0  |
-----------------------------------------------------------------|-------|-----------------|
Samlede pris: 0
```

vp(sim10,sim10,14,2,0,step,1000,0,2000).

```
Benyttet tid: 8.36000000000058
Found a solution with cost 2
Branch-and-bound timeout!

      m t o t f l s m t o t f l s m t o t f l s m t o t f l s | D A N | AV WS FØ EV sum |
-----------------------------------------------------------------|-------|-----------------|
Nr. 1: N S F F F A A F D F F F F F F N S F F D A A F F D F F F F | 3 4 2 | 0  0  0  0   0  |
Nr. 2: F N S F F D D F A F F F F F F N S F F D D F A A A F F F | 4 4 2 | 0  0  0  0   0  |
Nr. 3: F F N S F F F D F F F F F N N S F A F F F F F F F D D A A | 3 3 3 | 0  0  0  0   0  |
Nr. 4: F F F N S F F A F F F F F F D D F F A N N S F F F F F F | 2 2 3 | 0  0  0  0   0  |
Nr. 5: A F F D N S F F F A F F F F A F F D F F F F F F F F N N | 2 3 3 | 0  0  0  0   0  |
-----------------------------------------------------------------|-------|-----------------|
Nr. 6: F D F A F F F N S D F F A A F F N N S F F F F F F F D D | 4 3 3 | 1  0  0  0   1  |
Nr. 7: F F F F F A N N S F F F D A F F F F F F F N S F D F F F F F F | 2 2 3 | 0  0  0  0   0  |
Nr. 8: F F D F D F F F F F F A F D D F A F F F F F F N N N S A F F | 4 3 3 | 1  0  0  0   1  |
Nr. 9: F F A F F F F F F N N S D F F F F F F A F F F F D F N S F F | 2 2 3 | 0  0  0  0   0  |
Nr.10: D A F F F F F F F F F N N N S F F F D F F F F A F F F N S F | 2 2 3 | 0  0  0  0   0  |
-----------------------------------------------------------------|-------|-----------------|
Samlede pris: 2
(Delløsning: pris: 0, tid: 5)
```

vp(sim10,sim10,28,0,1,step,1000,0,2000).

```
Benyttet tid: 4.48000000000138
Found a solution with cost 2
Branch-and-bound timeout!

      m t o t f l s m t o t f l s m t o t f l s m t o t f l s | D A N | AV WS FØ EV sum |
-----------------------------------------------------------------|-------|-----------------|
Nr. 1: F F D D D A A F F F F F F F F F F F F A A F F N N S F F | 3 4 2 | 0  0  0  0   0  |
Nr. 2: F D N N S F F A F F F F F F F A F F F F F F A F F F D D D | 4 3 2 | 0  0  0  0   0  |
Nr. 3: N S A A F F F F F F F F F N N S F F F F F F D D D A F F F | 3 3 3 | 0  0  0  0   0  |
Nr. 4: A N S F F D D N N S A F F F F A F F F D D F F F F F F F F | 4 3 3 | 1  0  0  0   1  |
Nr. 5: F F F F F F F F F F N N N S F F D D A A F F F F F D F F F | 3 2 3 | 0  0  0  0   0  |
-----------------------------------------------------------------|-------|-----------------|
Nr. 6: D A F F N S F F F F F F F F F D F F D F N N S A F F A F F | 3 3 3 | 0  0  0  0   0  |
Nr. 7: F F F F F F F F D D F F F A A F F F N S F F N N S F F A A | 2 4 3 | 1  0  0  0   1  |
Nr. 8: F F F F F F F F F A A F D F F N N N S D F F F F F F F F F | 2 2 3 | 0  0  0  0   0  |
Nr. 9: F F F F F F N N S F D D A F F F F A F N S F F F F F F F F | 2 2 3 | 0  0  0  0   0  |
Nr.10: F F F F A F F F F F F F D D F F F F F F F F F A F N N N | 2 2 3 | 0  0  0  0   0  |
-----------------------------------------------------------------|-------|-----------------|
Samlede pris: 2
```

vp(sim10,sim10,14,2,1,step,500,0,2000).

```
Benyttet tid: 2.8799999999992
Found a solution with cost 3
Branch-and-bound timeout!

      m t o t f l s m t o t f l s m t o t f l s m t o t f l s | D A N | AV WS FØ EV  sum |
-----------------------------------------------------------------|-------|-----------------|
Nr. 1: D A F F N S F F F F F F F F F F A A F F F D D F N S F | 3 3 2 | 0  0  0  0   0  |
Nr. 2: F F D F F F F D F A F N S F A N S F F F F D F F F F A A | 3 4 2 | 0  0  0  0   0  |
Nr. 3: F F F F A F F F F N S F N N S A F D D F F F F F D D F F | 4 2 3 | 1  0  0  0   1  |
Nr. 4: F D A F F F F F F F N S A A F F A F F F F F F N N S D D | 3 4 3 | 1  0  0  0   1  |
Nr. 5: F F F F F A A F N S D F F F F F D F F F F F F F F N N | 2 2 3 | 0  0  0  0   0  |
-----------------------------------------------------------------|-------|-----------------|
Nr. 6: N S F F D F F A F F F D F F F F F F F N N S F A A A F F | 2 4 3 | 1  0  0  0   1  |
Nr. 7: F F N N S F F F F D F F D D F F N S F F F A A F F F F F | 3 2 3 | 0  0  0  0   0  |
Nr. 8: F F F A F F F F F D F F A F F D D F F N S F N N S F F F F | 3 2 3 | 0  0  0  0   0  |
Nr. 9: F N S F F D D N S F A F F F N S F F F A A F F F F F F F | 2 3 3 | 0  0  0  0   0  |
Nr.10: A F F D F N N S A F F F F F F F F N S D D F F F F F F F | 3 2 3 | 0  0  0  0   0  |
-----------------------------------------------------------------|-------|-----------------|
Samlede pris: 3
```

D.12 hol13u8

vp(hol13u8,hol13,56,0,0,step,1000,0,2000).

```
Benyttet tid: 71.5599999999999
Found a solution with cost 32
Branch-and-bound timeout!

      m t o t f l s m t o t f l s m t o t f l s m t o t f l s m t o t f l s
-------------------------------------------------------------------------------
Nr. 1: F F F D F N N S D D D F F F D D D D F N N S D F F F F F F F F F A A F F F D D F F
Nr. 2: F F F F D F F D D D D F N N S D D D F F F F D F F F N N S F F F F F F F F F F A A
Nr. 3: F F A F D A A F A A A A F F F F F F D A A F F D D D F F D D D D F N N S F F F D F F
Nr. 4: F F F F F A A F A A A A F F F F F F D A A F F D D D F F D D D D F F F F F F F N N
Nr. 5: N N S F F D D N N S F D F F F A A F F D D F A A F F F F F F F F D A A F D D F F F F
-------------------------------------------------------------------------------
Nr. 6: F F N N S D D F F N N S F F F F A A F D D F F A A A F F A A A F F D D F F F F F F F
Nr. 7: F D F F F F F F F F F F A A N N N S F F F N S F F F A A F F F F D F F D D D F F A A
Nr. 8: A F F F F F F F F F F F D A A F A F N S F F F N S F F A A N N S A F F F F A A F F D D
Nr. 9: F F F F F F F F F F F F D D F F F A A F F F F N N S D D F A N N S F F F A A F F D D
Nr.10: D A F F F F F A F F F F D D A F F F A F F D A F F F D D A F A A A F F N N N N S F F
-------------------------------------------------------------------------------
Nr.11: F F D D N S F A F F F F F F A F F F N S F A F F A A F F F F F F F D D A F F A A F F
Nr.12: D D D A A F F D F F F N S F D F F F F F F A F F F N S F F F F F A F F A F F A N S F
Nr.13: A A A A A F F F F F F F F F F F F F F F D F F F F F F F F F F N S F D F F D A F F
-------------------------------------------------------------------------------
                  m t o t f l s m t o t f l s |  D  A  N | AV WS FØ EV  sum |
-------           ----------------------------|----------|------------------|
Nr. 1:             F F F F F A A A A A A F F F | 11  8  4 |  2  0  0  0   2  |
Nr. 2:             A A A A F F F D F A A F A A | 10 10  4 |  2  0  0  0   2  |
Nr. 3:             F F F F D N N S F F F F F F | 11  9  4 |  2  0  0  0   2  |
Nr. 4:             S F F F F F F F A F D F N N |  9  9  4 |  0  0  0  0   0  |
Nr. 5:             F F F F F A A F F F F A F F |  8  9  4 |  0  0  0  0   0  |
-------           ----------------------------|----------|------------------|
Nr. 6:             F F F F F D D F D F F A F F |  9  9  4 |  0  0  0  0   0  |
Nr. 7:             F D D D D F F F F F F F A A |  9  8  4 |  0  0  0  0   0  |
Nr. 8:             F F F F F F F F F D D D D D |  8  9  4 |  0  0  0  0   0  |
Nr. 9:             F A A A A F F F F F F D D D |  9  9  4 |  0  0  0  0   0  |
Nr.10:             N S D F F D D F F F F F F F |  9  9  5 |  0  0  0  0   0  |
-------           ----------------------------|----------|------------------|
Nr.11:             D N N N S F F D D D F F F F |  8  8  5 |  0  0  0  0   0  |
Nr.12:             A F F D A F F N S F F N S F |  6  8  5 |  2  0  0  0   2  |
Nr.13:             D D F F N S F A N N N S F F |  5  7  5 |  4  0  0  0   4  |
-------           ----------------------------|----------|------------------|
AV: 'Antal vagter'
WS: 'Weekendsammenlægning'
FØ: 'Fridagsønsker'
EV: 'Efterfølgende vagter'

Samlede pris: 12
Strategi: step
Benyttet tid: 1000.68
```

vp(hol13u8,hol13,28,2,0,step,500,0,2000).

```
Benyttet tid: 562.18
Found a solution with cost 21
Branch-and-bound timeout!

      m t o t f l s m t o t f l s m t o t f l s m t o t f l s m t o t f l s m t o t f l s
-------------------------------------------------------------------------------------
Nr. 1: F F F D F N N S D D D F F F F D F F F A A A A F F F F F F D D F D F N N S F F F F F F
Nr. 2: A F F F F F F F D A A F F N N S D D D F F F F F D F F F N N S D D F D F F A F F F F A A
Nr. 3: A F F F F F A A F D D D D F F F A A F F N N S F D F F F F F A A F D D F F F D F F F N N
Nr. 4: F F F F F F A A F A A F D F F F F F D D D F F F F F D F F A A N N S F A F F F A F D D F F
Nr. 5: N N S F F D D F F F F F A F F D A F F D A A F F F F D A F F F F F D A F A A F D D F F F F
-------------------------------------------------------------------------------------
Nr. 6: F F N N S D D F F F A A F F F F F A A F D D F F F F F A F F F F N N S A A F F D D D F F
Nr. 7: F F D D D F F N N S F F A A F F F F F F F F D D F F F A A F A A F F F F N N S F F A A
Nr. 8: F A F F F F F F F F N N S A A F F F F A A F F F F F F D D D D F F F A A F F F F N N S D D
Nr. 9: F F F F F F F F A F F A F D D N N S F A F F A F F F F D D D F F A F F F F F D A A F F D D
Nr.10: F D F F F D F F A F F F F F D D A F N N S F F D A A A F F F F F F F F F D D F F A A A F F
-------------------------------------------------------------------------------------
Nr.11: F F D A N S F D F F F F F F F A F F F F D D N S A A F F F F F F F F F D D F F F A A F F
Nr.12: D D A A A F F F F F F F N S F D F F F F F F F F F F N S N S F D F F F F F F F A F F F N S F
Nr.13: D A A F A F F F F F F F F F F F F F F N S F F N S N S F F A F F F N S F D F F F F F F F
-------------------------------------------------------------------------------------
                    m t o t f l s m t o t f l s |  D  A  N | AV WS FØ EV  sum |
-------             ----------------------------|---------|-----------------|
Nr. 1:               F F D D F A A A A A F F F F | 10  9  4 |  1  0  0  0   1  |
Nr. 2:               A F F F F F F F D F F F F A A |  9  9  4 |  0  0  0  0   0  |
Nr. 3:               S F F F F F F F F D F F F A A |  9  9  4 |  0  0  0  0   0  |
Nr. 4:               F D F F F N N S F D A F F F |  9  9  4 |  0  0  0  0   0  |
Nr. 5:               F F F F D F F F F F F F N N |  9  8  4 |  0  0  0  0   0  |
-------             ----------------------------|---------|-----------------|
Nr. 6:               F D F F D A A F F F F F F F |  9  9  4 |  0  0  0  0   0  |
Nr. 7:               F F F F A F F F F F D D D D |  9  9  4 |  0  0  0  0   0  |
Nr. 8:               F A F F F F F F F F F D D D |  9  8  4 |  0  0  0  0   0  |
Nr. 9:               N N S A F F F F F F D A F F |  9  9  4 |  0  0  0  0   0  |
Nr.10:               F F N N S D D F F F N S F F |  9  8  5 |  0  0  0  0   0  |
-------             ----------------------------|---------|-----------------|
Nr.11:               D A A F F D D N N N S F F F |  9  8  5 |  0  0  0  0   0  |
Nr.12:               A F A A A F F D D D F N S F |  7  8  5 |  1  0  0  1   2  |
Nr.13:               D F D D N S F A A A A A F F |  5  9  5 |  3  0  0  1   4  |
-------             ----------------------------|---------|-----------------|
AV: 'Antal vagter'
WS: 'Weekendsammenlægning'
FØ: 'Fridagsønsker'
EV: 'Efterfølgende vagter'

Samlede pris: 7
Strategi: step
Benyttet tid: 1001.94


(Delløsning: pris: 4, tid: 23)
```

vp(hol13u8,hol13,14,2,0,step,500,0,2000).

```
Benyttet tid: 1148.32
Found a solution with cost 3
Branch-and-bound timeout!

      m t o t f l s m t o t f l s m t o t f l s m t o t f l s m t o t f l s
-------------------------------------------------------------------------------
Nr. 1: N S F F D A A F D F F F F F N S F F D A A F F D F F F F F N S F F D A A F F D F D F F
Nr. 2: F N S F F A A F D D F D F F A N S F F A A F D F F D F F F N S F F A A F D F D F F F
Nr. 3: F F N S F D D F A A F F F F F F N S F D D A A F F D F F F F N S F D D A A F F A F F
Nr. 4: F A F N S D D F A A F F F F F F A N S D D F F D A F F F F F A N S D D F F F F A F F
Nr. 5: A F F F D F F N S D F F A A F D D F F F F F N S F F F A A F D D F F F F N S D F F A A
-------------------------------------------------------------------------------
Nr. 6: A F F D F F F D N S F F A A F D D F F F F F N S F F A A F D D F D F F F N S F F A A
Nr. 7: F F F A A F F F F N S F D D D F A A F F F F F N S F D D F F A A A F F F F N S F D D
Nr. 8: F F F F A F F A F F N S D D F A F A F F F F F F N S D D A A F A F F F F F F N S D D
Nr. 9: F F D A N S F A F F D F F F A A F D A F F D F F D N S F A A F D F F F F F F F N S F
Nr.10: D D A F F F F D F F A N S F F F F D A F F A F F D F N N S F F D F F F A A F D F N N
-------------------------------------------------------------------------------
Nr.11: F F D D F F F F F F A A N N S F F F D F F D F A A A F F F F F F F N N S D F F D F F
Nr.12: D D A F F F F F F F D A F F F F F F N N N S A F F F F F D F F F N S F D F A A F F F
Nr.13: F A F F F N N S F F F D F F D F F F F F F F D A F A F F D F F F A F F D F A A F F F
-------------------------------------------------------------------------------
                    m t o t f l s m t o t f l s |  D  A  N | AV WS FØ EV  sum |
-------             ----------------------------|----------|------------------|
Nr. 1:               N S F F D A A F F F F D F F |  9  8  4 |  0  0  0  0   0  |
Nr. 2:               F N S F F A A F D D F F F F |  9  9  4 |  0  0  0  0   0  |
Nr. 3:               F F N S F D D A F F A F F F |  9  9  4 |  0  0  0  0   0  |
Nr. 4:               F F A N S D D F F F F A F F |  9  9  4 |  0  0  0  0   0  |
Nr. 5:               F F D F F F F N S D F F A A |  9  9  4 |  0  0  0  0   0  |
-------             ----------------------------|----------|------------------|
Nr. 6:               F F D F F F F F N S F F A A |  8  9  4 |  0  0  0  0   0  |
Nr. 7:               A F A F F F F F F N S F D D |  9  9  4 |  0  0  0  0   0  |
Nr. 8:               A A F F F F F F D F N S D D |  9  9  4 |  0  0  0  0   0  |
Nr. 9:               F A F D D F F F F F D N S F |  9  8  4 |  0  0  0  0   0  |
Nr.10:               S D F F A F F A F F D F F F |  9  8  5 |  0  0  0  0   0  |
-------             ----------------------------|----------|------------------|
Nr.11:               F D F A A F F D A F F F N S |  8  8  5 |  0  1  0  0   1  |
Nr.12:               D F F A F F F D A A F D F N |  8  8  5 |  0  1  0  0   1  |
Nr.13:               D F F D N N N S F A A A F F |  7  9  5 |  1  0  0  0   1  |
-------             ----------------------------|----------|------------------|
AV: 'Antal vagter'
WS: 'Weekendsammenlægning'
FØ: 'Fridagsønsker'
EV: 'Efterfølgende vagter'

Samlede pris: 3

1. del: pris: 2, tid:  1
2. del:       2       66
3. del:       0       20
4. del:       3      124
```

Litteratur

[1] Arbejdsministeriet (1999),
Bekendtgørelse af lov om arbejdsmiljø, LBK nr 784 af 11/10/1999,
http://www.retsinfo.dk/_GETDOCI_/ACCN/A19990078429-REGL.

[2] Baptiste, P., Le Pape, C. og Nuijten, W. (2001)
Constraint-Based Scheduling, Applying Constraint Programming to Scheduling Problems,
Kluwer Academic Publishers Group.

[3] Barták, R. (1998)
On-Line Guide to constraint programming,
http://ktiml.mff.cuni.cz/ bartak/constraints/

[4] Barták, R. (1999)
Constraint Programming: In Pursuit of the Holy Grail,
Charles University, Faculty of Mathematics and Physics, Department of Theoretical Computer Science.

[5] Blackburn, P., Bos, J., Striegnitz, K. (2001)
Learn Prolog Now!,
http://www.coli.uni-sb.de/ kris/learn-prolog-now/html/prolog-notes.ps.gz

[6] Cheng, B. M. W., Lee, J. H. M. og Wu, J. C. K. (1997)
A Nurse Rostering System Using Constraint Programming and Redundant Modeling,
IEEE Transactions on information technology in biomedicine, Vol. 1, no. 1, march 1997.

[7] Dansk Sygeplejeråd, Kommunernes Landsforening (1999)
Arbejdstidsaftale for syge- og sundhedsplejersker, 1999 - KL,
http://www.sygeplejersken.dk/job/tekstlink.asp?id=1500000&menu=194000

[8] Dansk sygeplejeråd, Kommunernes Landsforening (1999)
Primæroverenskomst,
http://www.sygeplejersken.dk/job/tekstlink.asp?id=1450000&menu=194000

[9] GAMS Development Corporation,
GAMS home page,
http://www.gams.com/Default.htm

[10] IC Parc at Imperial College,
The ECL^{i}PS^e Constraint Logic Programming System,
http://www.icparc.ic.ac.uk/eclipse/

[11] Lustig, Irvin J., Puget, Jean-François
Program Does Not Equal Program: Constraint Programming and Its Relationship to Mathematical Programming,
Interfaces 31: 6 November-December 2001 (pp. 29-53).

[12] Marriott, K. og Stuckey, J. P. (1998)
Programing with Constraints, An Introducion,
The MIT Press.

[13] Schimpf, J., Shen, K. og Wallace, M. (2002)
Tutorial on Search in ECL^{i}PS^e,
Imperial College London.

[14] Sterling, L. (1994)

The art of Prolog: Advanced programming techniques The MIT Press.

[15] Weil, Georges m. fl. (1995),
Constraint Programming for Nurse Scheduling,
IEEE Engeniering in medicine and biology, July/August 1995.

[16] Wallace, Mark (1995),
Constraint Programmig,
http://www-icparc.doc.ic.ac.uk/eclipse/reports/handbook/handbook.html Imperial College London.