



AARHUS  
UNIVERSITET  
INSTITUT FOR MATEMATIK

# RUTEPLANLÆGNING I HJEMMEPLEJEN

HOME HEALTH CARE ROUTING

BACHELORPROJEKT

LAURA HORSFALL FOLMER - 201905054

TONY LUNG CA HIN - 201905921

15. JUNI 2022

VEJLEDER: JENS LYSGAARD

## **Resumé**

The purpose of this report is to examine vehicle routing in home health care settings. It focuses on daily planning of a team of health care workers, based on real scenarios and real needs of the citizens that receive home health care in Denmark. This paper presents known mathematical programs in order to understand how to implement solutions for scheduling home health care workers. A test dataset of 50 fictitious citizens that must be visited within a time window is used to examine formulations of asymmetrical vehicle routing programs. Solutions are visualised to identify benefits and issues with the models.

We consider solutions to precedence, synchronisation, time window and capacity constraints in visits, as well as account for health care worker's working hours and discuss how work breaks can be implemented. All presented mixed-integer program formulations are solved with CPLEX.

Our analysis shows that small test instances can be solved in reasonable times and the span of the time windows impacts the feasibility of solutions. Lastly, the different models are compared in terms of solution times and complexity of implementation.

# Indhold

<b>1</b>	<b>Introduktion</b>	<b>1</b>
1.1	Hjemmepleje . . . . .	1
1.1.1	Hjemmeplejens ruteplanlægning . . . . .	1
1.1.2	Borgernes behov . . . . .	2
1.1.3	Ruteplanlægning i hjemmeplejen . . . . .	2
1.2	Problemformulering . . . . .	3
1.3	Metodeafsnit . . . . .	3
1.3.1	Data . . . . .	3
1.3.2	Omkostninger . . . . .	4
1.3.3	Metode . . . . .	5
<b>2</b>	<b>Traveling Salesman Problem</b>	<b>6</b>
2.1	Eliminering af subture . . . . .	7
2.1.1	DFJ-SEC . . . . .	7
2.1.2	MTZ-SEC . . . . .	7
2.2	Anvendelse af TSP . . . . .	8
2.3	Branch and bound . . . . .	9
2.3.1	Illustrerende eksempel . . . . .	10
2.4	CPLEX . . . . .	11
<b>3</b>	<b>Vehicle Routing Problem</b>	<b>13</b>
3.1	Capacitated Vehicle Routing Problem . . . . .	13
3.1.1	Kapacitetsbegrænsninger . . . . .	14
3.1.2	Anvendelse af CVRP . . . . .	15
3.2	Capacitated Vehicle Routing Problem med Tidsvinduer . . . . .	17
3.2.1	MTZ . . . . .	17
3.2.2	Infeasible Path Elimination Constraints . . . . .	18
3.2.3	Anvendelse af CVRPTW1 . . . . .	19
3.2.4	Anvendelse af CVRPTW2 . . . . .	20
3.2.5	Betydning af tidsvinduerne's længde . . . . .	22
3.2.6	Tre-indeksering . . . . .	23
3.2.7	Synkronisering . . . . .	24
3.2.8	Anvendelse af tre-indeksering . . . . .	25
3.2.9	Pauser for medarbejdere . . . . .	28
<b>4</b>	<b>Resultater</b>	<b>29</b>

<b>5 Diskussion</b>	<b>30</b>
5.1 Valg og fravalg . . . . .	30
5.2 Implementering og løsning . . . . .	31
<b>6 Konklusion</b>	<b>32</b>
<b>Litteratur</b>	<b>33</b>
<b>Bilag</b>	<b>36</b>
Bilag 1: Datasæt . . . . .	36
Bilag 2: Jens Lysgaard 2021, Slides . . . . .	38
Bilag 3: Kodeoversigt . . . . .	57

# 1 Introduktion

*Denne opgave er lavet af Tony Lung Ca Hin og Laura Horsfall Folmer. Hele opgaven er udarbejdet og skrevet i fællesskab.*

## 1.1 Hjemmepleje

Hjemmepleje er en service, som gives til borgere i Danmark der enten permanent, fx pga. alder eller handicap, eller midlertidigt ikke kan tage vare på sig selv. I 2021 modtog 153.600 personer i Danmark hjemmehjælp og 140.100 personer modtog hjemmesygepleje (Danmarks Statistik 2022 [6]). Antallet af personer visiteret til hjemmehjælp er steget siden 2015, men samtidig er antallet af ældre i landet generelt også steget, det vil sige, andelen af hjemmehjælpsmodtagere er altså faldet. Derudover blev der i gennemsnit bevilget 497800 timers hjemmehjælp per uge i 2021, hvilket svarer til en gennemsnitlig hjælp på 3,2 timer om ugen per hjemmehjælpsmodtager. Hjemmepleje vil i denne opgave dække over både hjemmesygepleje og hjemmehjælp. Hjemmepleje kan fås enten privat eller gennem kommunen og dækker over et bredt spektrum af ydelser, heriblandt personlig pleje; hjælp til bad, toiletbesøg osv. og hjælp til medicinering, rengøring, indkøb eller andre praktiske gøremål.

### 1.1.1 Hjemmeplejens ruteplanlægning

Da hjemmehjælpen skal ud til borgerne i deres eget hjem, kræver det organisering og planlægning. En undersøgelse af Rambøll fra 2018 [26], kigger netop på planlægningen og tilrettelæggelsen af opgaver i hjemmeplejen. Ifølge undersøgelsen udgør såkaldte kørelister et vigtigt værktøj for planlægningen. Kørelisterne indeholder den enkelte helpers rute og er altså en plan over besøgene hos borgerne. Af hensyn til for eksempel, borgernes behov eller sygdom hos personalet, opdateres kørelisterne daglig af skemaplanlæggerne (Rambøll 2018, s. 4 [26]). Undersøgelsen viser at tilrettelæggelsen af kørelisterne hos de syv forskellige involverede kommuner, ikke er ensartet og således varierer kørelistemodellerne fra kommune til kommune. Der lægges i rapporten vægt på fordele og ulemper ved at planlægge kørelisterne på forskellige måder. Til planlægning af kørelisterne identificeres centrale hensyn:

- Krav til specifikke faglige kompetencer
- Geografiske afstande, med henblik på at reducere køretiden mest muligt
- Kontinuitet og sammenhæng i borgerens forløb
- Personlige relationer og præferencer blandt borgere og medarbejdere
- Forskellige krav til dokumentation

- Udvikling i borgers tilstand og plejebehov
- Plejeopgavens kompleksitet
- Overholdelse af overenskomster

For at kunne tage højde for de mange hensyn, har alle på nær én kommune i undersøgelsen en fuldtid eller deltidsplanlægger (Rambøll 2018, s. 7 [26]). Til udarbejdelse af kørelister, tager planlæggeren således hensyn til kravene baseret på erfaring og IT-systemer på daglig basis. Medarbejderne møder ind til morgenmøde, hvor de orienterer sig i kørelisterne og indbyrdes eventuelt bytter lister eller besøg. Medarbejderne tilpasser sig kørelisten og ændrer måske selv i ruten, hvis de for eksempel selv ved hvornår en bestemt borgers vil besøges, eller hvis de synes at en anden rute er bedre (Rambøll 2018, s. 13 [26]). I rapporten fra Rambøll fremgår det også, at det varierer, hvor meget information om kørselstider hver hjælper får på deres køreliste. Nogle får slet ingen tidspunkter, men blot en rækkefølge af borgere og en servicetid, mens det hos andre er meget tydeligt mellem hvert besøg, hvor lang tid der er sat af til kørsel mellem borgerne. Nogle får også kørselstider i blokke, det vil sige mellem nogle af besøgene. Udover kørselstiden tages der også højde for, hvornår på dagen en borgers har brug for hjælp - skal de for eksempel have hjælp til at komme ud af sengen, i bad eller have lavet aftensmad (Rambøll 2018 [26]). Hjemmeplejens ruteplanlægning er da en kompliceret opgave, og det er ikke entydigt hvordan opgaven løses.

### **1.1.2 Borgernes behov**

Der skal som nævnt tages højde for borgernes behov i ruteplanlægningen. I en undersøgelse af hjemmeplejen lavet af PwC for Videnscenter for Værdig Åldrepleje (2021 [24]) peger de ældre på, at de værdsætter at hjælperen kommer på et fast tidspunkt hver dag, da det giver forudsigelighed. Der nævnes også at nogle borgere har en oplevelse af at der ikke er afsat tilstrækkelig tid til kørsel mellem borgerne (PwC 2021, s. 10 [24]). Derudover gives der udtryk for at borgeren ønsker at få tilknyttet et færre antal faste hjælpere, da de er dygtige til at opfange borgers trivsel og behov, samt medfører et mere personligt og familiært forhold til hjælperen (PwC 2018, s. 10 [24]). Borgernes oplever ifølge undersøgelsen også at hjælperne generelt har travlt, og ønsker at der er mere tid til for eksempel at sidde ned og snakke. Hjælperne giver selv udtryk for denne travlhed og det kan betyde at nogle ældre ikke udtrykker deres behov overfor dem (PwC 2018, s. 24 [24]).

### **1.1.3 Ruteplanlægning i hjemmeplejen**

Ruteplanlægning i hjemmeplejen betyder altså en stor del for både samfundets ressourceforbrug, hjælpernes arbejdsgang samt borgernes tilfredshed. En optimering af ruteplanlægningen og dermed kørelisterne for hjælperne kan være med til både at spare penge hos kommunen eller firmaet og spare transporttid hos medarbejderne. For at optimere ruteplanlægningen vil vi forsøge, med

lineær programmering, at bestemme de bedste ruter for både en enkelt og flere hjemmehjælperne. Hvilken rute der er bedst kan afhænge af mange ting, men vi vil forsøge at finde de bedste ruter ved at minimere afstandene, som hjælperne skal køre på en dag. Minimering af rejsetid er et kendt problem og bruges ikke kun indenfor hjemmeplejen. Derfor findes der mange kendte modeller til opstilling af lignende problemstillinger. Denne opgave søger ikke at tage højde for alle nævnte problematikker og udfordringer indenfor ruteplanlægning i hjemmeplejen, men vil forsøge at minimere kørselstiden og samtidig forsøge at løse nogle af de nævnte problematikker.

## 1.2 Problemformulering

Formålet med denne bacheloropgave er at undersøge ruteplanlægningsproblemer med udgangspunkt i tænkte scenarier i hjemmeplejen. Disse scenarier opstilles matematisk og løses med matematisk programmering. Scenarierne opstilles med henblik på at afspejle virkeligheden.

### Problemformulering:

- Hvordan kan ruteplanlægning optimeres i hjemmeplejen i henhold til borgernes servicebehov, og hvilke kendte metoder og modeller kan anvendes til opstilling af forskellige scenarier i hjemmeplejen?

### Problemstillinger:

- Hvilke behov har borgerne og hvordan opfyldes de?
- Hvordan kan praktiske problemstillinger indenfor hjemmeplejen modelleres med matematisk modellering?
- Hvordan fungerer løsningen af de forskellige modeller i forhold til sværhedsgrad af implementering og løsning?

## 1.3 Metodeafsnit

### 1.3.1 Data

For at kunne sammenligne de forskellige modeller og løsninger bedst, er der brugt det samme datasæt gennem hele opgaven. Der er til formålet blevet lavet et datasæt, *Borgerdata*, til at repræsentere en liste med en række borgere og deres tilhørende information. Et udsnit af datasættet kan ses i Tabel 1 (se hele datasættet i Bilag 1).

Navn	Adresse	Besøgsvindue	Servicetid
Depot	(30,30)	00:00-99:99	00:00
1	(29,56)	07:30-08:00	00:30
2	(10,1)	11:00-13:00	00:20
3	(2,2)	16:00-17:00	00:20
:	:	:	:

Tabel 1: Udsnit af datasættet *Borgerdata*.

*Borgerdata* har i alt 51 observationer, det vil sige hjemmeplejens kontor, her kaldet Depot og halvtreds borgere. Den første kolonne, 'Navn', angiver nummeret på borgeren. Adresserne i datasættet er tilfældigt genererede koordinater, hvor begge værdier er et tal mellem 0 og 60. Depotet er placeret i midten, det vil sige, på koordinatet (30,30).

Besøgsvinduet, det vil sige, tidsrummet hvori hjælperen skal ankomme hos borgeren, er konstrueret manuelt og fordeler sig i tidsrummet 7-21. Længderne på disse spænder mellem 30 minutter og 5 timer.

Servicetiden er varigheden af en opgave og er også konstrueret manuelt og spænder mellem 10 minutter og 1 time. Servicetiderne er valgt ud fra beskrivelserne i artiklen *Arbejdssdag på 10 timer i hjemmeplejen* fra magasinet Sygeplejersken (Røge 2017 [28]). Forfatteren af artiklen er hjemmesygeplejerske og servicetiden hos den enkelte borger varierer alt efter, hvilken behandling borgeren skal have. Eksempelvis har forfatteren af artiklen 60 min. hos borger 9 til at behandle tre komplicerede bensår og skylle en byld, hvor hun hos borger 1 har 15 min. til at give en blodfortyndende injektion.

Datasættet forsøger så vidt muligt at afspejle et virklig scenarie mht. besøgstider, afstande og antal borgere pr. hjælper. I artiklen har hjemmesygeplejersken 11 borgere på sin køreliste og vikaren har 8, og det forventes da at fem fastansatte vil være nok til at dække besøgene. Dog er det et nedskaleret eksempel, da et typisk plejecenter vil have både flere borgere og flere medarbejdere. For eksempel, havde de ifølge en tilsynsrapport fra 2022 i Odder, 320 patienter tilknyttet hjemmeplejen med 35 ruter i dagvagt og 12 i aftenvagt (Styrelsen for Patientsikkerhed 2022 [29]), men information om, hvorvidt alle borgere bliver besøgt hver dag gives ikke. *Borgerdata* kan dog betragtes som et udsnit af kommunen, hvor et team af hjemmehjælpere er sat på netop de borgere.

### 1.3.2 Omkostninger

Omkostningerne ved at køre fra et punkt til et andet regnes som den euklidiske afstand mellem punkterne, det vil sige, den direkte lineære afstand mellem punkterne (Laporte 1992 [16]). Dette skal repræsentere den omkostning der er forbundet med kørslen fra den ene adresse til den anden. Omkostningerne kan her være kørselsafstand eller kørselstid. Vi antager, at hastigheden konstant er 1 for alle køretøjer (Bard et al. 2002 [3]). Det medfører, at kørselsafstand er lig kørselstiden.

For alle modeller i det følgende, vil omkostningerne  $c_{ij}$  for hver afstand mellem to punkter,  $i$  og  $j$ , udregnes og danne en symmetrisk omkostningsmatrix  $C$ . Den euklidiske afstand i to dimensioner for to punkter med dertilhørende koordinatsæt  $p = (p_1, p_2)$ ,  $q = (q_1, q_2)$  beregnes således:

$$c_{pq} = \sqrt{(q_1 - p_1)^2 + (q_2 - p_2)^2}$$

Da vi benytter euklidiske afstande, er trekantsuligheden opfyldt (Laporte 1992 [16]). Det vil sige, for tre punkter  $p, q, r$  er det opfyldt at,

$$c_{pq} + c_{qr} \geq c_{pr}.$$

Punkterne er tilfældigt genererede og derfor benyttes der blot euklidisk afstand som mål for kørselstiden. Havde virkelig data været tilgængelig, kunne kørselstiden beregnes mere komplekst på de faktiske ruter som hjælperne ville køre. Til implementering omregnes besøgsvinduets timer og minutter til antal minutter fra kl 00:00.

### 1.3.3 Metode

Gennem opgaven og til at besvare problemformuleringen, opstilles matematiske modeller for de forskellige problemstillinger indenfor hjemmeplejen som projektet har til hensigt at undersøge. Opgaven er altså primær teoretisk, men med klart fokus på at repræsentere virkeligheden så vidt muligt ved at tage udgangspunkt i et konkret scenarie med hjemmeplejen.

Der er indenfor fagets metode gjort brug af Pids modelleringsprincipper (Pidd 1999 [22]). Når matematiske modeller opstilles, skal man være påpasselig med ikke at overkomplificere problemet. Dette kan lede til fejl i programmeringen og ligeledes komplificere fortolkningen og bekræftelsen af løsningen. Opgaven er derfor struktureret således, at modellerne udvikler sig gradvist i forhold til kompleksitet. I den første model, der opstilles, er der kun inkluderet de mest simple og nødvendige elementer i forhold til modellens formål, hvorefter mere komplicerede elementer tilføjes.

Kurset *Modellering og løsning af optimeringsproblemer* fra foråret 2021 med professor Jens Lysgaard er brugt som afsæt for denne opgave. Her blev blandt andet Traveling Salesman, Branch and Bound, Vehicle Routing Problem og Capacitated Vehicle Routing Problem berørt. Der bliver derfor også i denne opgave opstillet lineære modeller og gjort brug af eksakte metoder til løsning af optimeringsproblemerne.

Til at implementere Mixed-Integer-Program (MIP) modellerne er der brugt Python-pakken PuLP, som formulerer dem i LP-format (PuLP 2021 [23]). Disse løses af en solver, og der er her brugt CPLEX solveren, som er en udbredt kommerciel software lavet af IBM. CPLEX bliver berørt igen senere i opgaven. Når der henvises til en '.py'-fil, findes filen i vedhæftet zip-fil. En oversigt over mappestrukturen kan ses i Bilag 3. Filen functions.py indeholder hjælpefunktioner, som er brugt i de fleste programmer. Blandt andet funktionen til beregning af omkostningsmatricen, omregning af klokkeslæt og en funktion til indlæsning af data.

## 2 Traveling Salesman Problem

Det første problem som opstilles er Traveling Salesman Problemet (TSP). Traveling Salesman problemet optrådte allerede i 1937 i forbindelse med ruteplanlægning af skolebusser (Dantzig et al. 1954 [7]). Oprindeligt består problemet af en handelsrejsende, der skal besøge en række byer én gang og vende tilbage til sit udgangspunkt i den rækkefølge, der minimerer den totale afstand. TSP kan benyttes til at opstille det basale problem indenfor hjemmeplejen, hvor én hjælper skal besøge en række borgere i den rækkefølge, som minimerer kørselstiden.

Nedenfor er opstillet en model for TSP. Formuleringen er lavet med inspiration fra Laporte (1992 [16]). I det følgende er  $G = (V, A)$  en graf, hvor  $V = \{0, \dots, n\}$  er punkter og  $A$  er sættet af kanter mellem punkterne i grafen.  $C$  er en matrix, hvor hver indgang  $c_{ij}$  er omkostningen tilknyttet kanten fra  $i$  til  $j$  og  $c_{ii} = \infty$ . I TSP er omkostningerne symmetriske, det vil sige, for alle  $i, j \in A$ ,  $c_{ij} = c_{ji}$ . Den følgende model er dog asymmetrisk, idet omkostningerne  $c_{ij}$  kan være forskellig fra  $c_{ji}$ , men kan også benyttes ved symmetriske omkostninger. Beslutningsvariablene  $x_{ij}$  kan fortolkes således:

$$x_{ij} = \begin{cases} 1 & \text{hvis kanten } (i, j) \in A \text{ bliver brugt på ruten} \\ 0 & \text{ellers.} \end{cases}$$

**Optimeringsproblem med begrænsninger, TSP1:**

$$\text{minimer } \sum_{(i,j) \in A} c_{ij} x_{ij} \quad (1)$$

$$\text{med bibetingelser } \sum_{j \in V} x_{ij} = 1, \quad \text{for alle } i \in V \quad (2)$$

$$\sum_{i \in V} x_{ij} = 1, \quad \text{for alle } j \in V \quad (3)$$

$$+ \text{ subtur elimineringsbegrænsninger,} \quad (4)$$

$$x_{ij} \in \{0, 1\}, \quad \text{for } (i, j) \in A \quad (5)$$

I objektfunktionen (1) minimeres summen af omkostningerne,  $c_{ij}$ , for alle kanter, der bliver brugt i en optimal løsning. Bibetingelse (2) og (3) sørger for, at hver by bliver besøgt og forladt af den handelsrejsende én gang, altså går der kun én kant ind og én ud af hvert punkt. Bibetingelse (4) eliminerer subture - dvs. eliminerer kredse, som ikke indeholder alle  $n$  byer. Uden disse ville der findes mange løsninger hvor (2) og (3) er opfyldt, men hvor ruten ikke vil være sammenhængende. Bibetingelse (5) tvinger  $x_{ij}$  til at være en binær variabel.

## 2.1 Eliminering af subture

En subtur er i TSP en kreds, som ikke går igennem alle  $n$  punkter. Uden eliminering af subture, vil modellen finde en løsning, hvor alle punkter er forbundet men ikke nødvendigvis i én stor kreds, da det typisk giver en højere omkostning. Derfor elimineres subturene ved hjælp af begrænsninger kaldet 'Subtour Elimination Constraints' (SEC). Der findes flere forskellige metoder til at eliminere disse subture. De to mest kendte bliver fremlagt herunder og brugt til implementering.

### 2.1.1 DFJ-SEC

Dantzig, Fulkerson og Johnson (1954 [7]) bemærkede at mængden af punkter i en subtur  $S$  består af  $|S| < n$  punkter. Summen af beslutningsvariablene  $x_{ij}$  der indgår i  $S$ , vil da også være  $|S|$ . For at udelukke subturen fra løsningen kan man da introducere følgende begrænsning (Dantzig et al. 1954):

$$\sum_{i,j \in S} x_{ij} \leq |S| - 1$$

For at udelukke alle mulige subture er det nok at tilføje:

$$\sum_{i,j \in S} x_{ij} \leq |S| - 1, \quad \text{for } S \subset V, 2 \leq |S| \leq n - 1 \tag{6}$$

hvor  $V$  er alle punkter. Kardinaliteten for  $S$  er minimum 2 da subture af længde 1 er udelukket af (2) og (3), hvilket også medfører at subture af længde  $n$  (grundet 0-indeksering) er udelukket (Laporte 1992 [16]). Disse begrænsninger kaldes i litteraturen DFJ-SEC.

### 2.1.2 MTZ-SEC

MTZ-begrænsninger er opkaldt efter Miller, Tucker og Zemlin (1960 [19]). Der introduceres variablene  $u_i$  for  $i = 1, \dots, n$  samt begrænsningerne:

$$u_i - u_j + px_{ij} \leq p - 1 \quad \text{for } i, j = 1, \dots, n, i \neq j \tag{7}$$

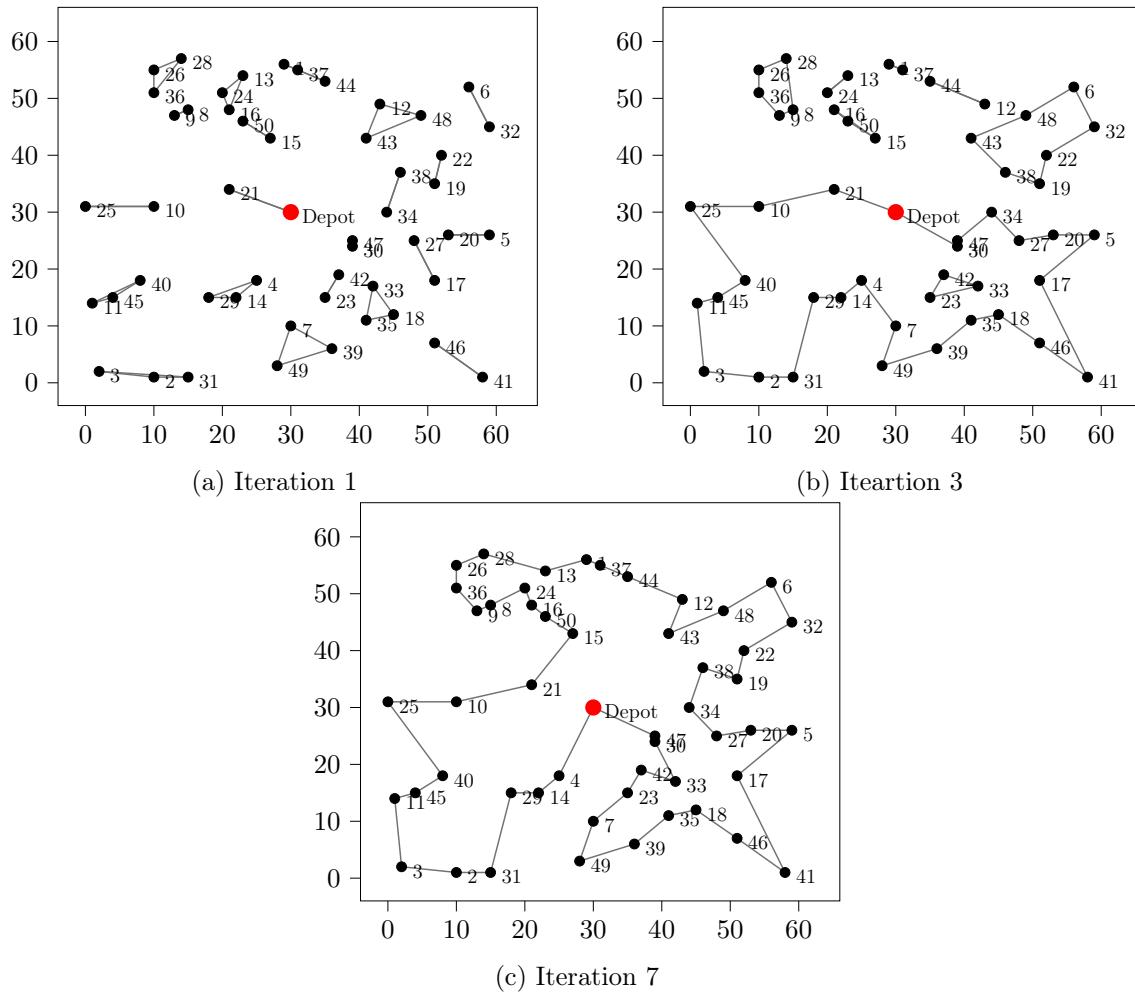
som udelukker subture, længere end  $p$  og hvor  $p$  er en øvre begrænsning på antallet af punkter i besøgt en rute. I artiklen af Miller, Tucker og Zemlin (1960 [19]) tillades det med  $p$ , at den handelsrejsende går igennem startpunktet flere gange, så for at det kan være en begrænsning til TSP, skal  $p = n$ . Variablene  $u_i$  fungerer som en indikatorvariabel, der tæller et punkts placering på ruten, udover startpunktet 0. Det kan altså vises, at

$$x_{ij} = 1 \Rightarrow u_j = u_i + 1.$$

Forskellen mellem de to typer af SEC er at antallet af DFJ-SEC vokser eksponentielt med antallet af punkter, mens MTZ-SEC kun vokser polynomisk (Bektas 2006 [4]). MTZ formuleringen er dog svagere end DFJ formuleringen (Laporte 1992 [16]).

## 2.2 Anvendelse af TSP

For at løse TSP1 med (6), er der implementeret en løsning i python, hvor (6) tilføjes iterativt, for hver subtur i en heltalsløsning. Konkret er det gjort ved først at løse TSP1 og derefter indsætte, en kant  $(i, j)$  hvis  $x_{ij} = 1$ , ind i en orienteret graf, ved hjælp af pythonpakken *NetworkX* (2021 [20]). I den pakke er implementeret, Johnsons Algoritme (Johnson 1975 [13]), til at finde alle cykler i grafen, og dermed også alle subture. Hver gang der fås en løsning kaldes det her en iteration. Første iteration uden SEC ses på 1a. Så længe en cykels længde er mindre end  $n$ , er det en subtur og (6) tilføjes. Programmet stoppes hvis løsningen er den samme, som forrige iteration. Dette sker kun hvis der ikke er en subtur, da den ellers vil blive udelukket fra løsningen og dermed vil næste iteration være forskellig. Løsningen er da brugbar og optimal når programmet stopper, hvilket sker efter 8 iterationer, og ses på 1c. Bemærk at der for eksempel på 1a er en cykel mellem borger 6 og borger 32, men man kun kan se en kant i figur 1. Pythonkoden for programmet og som også laver plots findes i `TSP_DFJ.py`.



Figur 1: Iteration 1,3 og 7 ud af 7 for TSP med DFJ-SEC

Metoden er inspireret af Pferschy og Stanek (2016 [21]), hvor SEC, genereres iterativt fra hel-talsløsninger, i stedet for at inkludere dem eksplisit i modellen. Da solverne som CPLEX, samt beregningskraft i moderne computere er blevet væsentlig hurtigere, kan det være en nem løsnings-metode. Figur 1a viser optimalløsningen til TSP uden SEC. Løsningen med MTZ-begrænsningen (7) giver samme løsning som Figur 1c, og findes i `TSP_MTZ.py`. Problemet er løst som en orien-teret graf, men da omkostningerne  $c_{ij} = c_{ji}$  er symmetriske, er rutens retning i dette tilfælde ligegyldig.

### 2.3 Branch and bound

Ruteplanlægning er typisk karakteriseret af heltalsløsninger og et af de mest succesfulde algoritmer til løsning af TSP er branch and bound-algoritmen (Williams 2013 [31]). Branch and bound-algoritmen blev først introduceret af Dantzig, Fulkerson & Johnson (1954 [7]) i forbindelse med løsning af TSP, men er sidenhen blevet en generel metode til at løse heltalsproblemer

(Balas og Toth i Lawler et al. 1985 [1]). Little et al. (1963 [18]) navngav algoritmen i år 1963 til 'branch and bound', hvor de introducerede deres algoritme. Deres algoritme kræver dog ikke nødvendigvis, at  $c_{ij} = c_{ji}$ , altså at omkostningerne er symmetriske, og algoritmen virker dermed både for TSP og det asymmetriske TSP (ATSP).

Generelt er idéen med Branch and Bound at generere delmængder af det brugbare sæt af løsninger (kaldet *branching*) og, for et minimeringsproblem, beregne en nedre grænse for omkostningerne for hver af delproblemerne (kaldet *bounding*). En nedre grænse kan fremkomme ved at relaksere problemet. En løsning, hvor alle variable er heltallige, kaldes brugbar og hvis dens totale omkostning samtidig er mindre eller lig med den nedre grænse for alle andre delmængder, er løsningen nu optimal for minimeringsproblemet.

Der findes forskellige metoder til både branching og bounding. Et eksempel på en bounding metode er LP-relaksation. Her løses et heltalsproblem (ILP-problem) som et LP-problem ved at fjerne heltalskravet for variablene. Der kan da foretages branching på de ikke-heltallige variable. Såfremt løsningen udelukkende består af heltallige variable, er løsningen som nævnt brugbar, og optimal.

Eksempelvis for en variabel  $x$  med ikke-heltallig optimalværdi  $\tilde{x}$ , branches der på denne variabel ved at dele problemet op i to delproblemer og tilføje hhv. begrænsninger  $x \leq \lfloor \tilde{x} \rfloor$  og  $x \geq \lceil \tilde{x} \rceil$  (Lysgaard, slide 19, Bilag 2). Undervejs når der branches ud af de forskellige delproblemer, vedligeholdes et søgeræ, hvor forskellige søgerestrategier, bl.a. Depth-First Search (DFS) eller Best Bound Search (BBS), kan bruges (Lysgaard, slide 16, Bilag 2).

### 2.3.1 Illustrerende eksempel

Det følgende er et eksempel på hvordan branch and bound kan benyttes til manuelt at løse et heltalsproblem. Eksemplet laves for at illustrere fremgangsmåden bag branch and bound algoritmen.

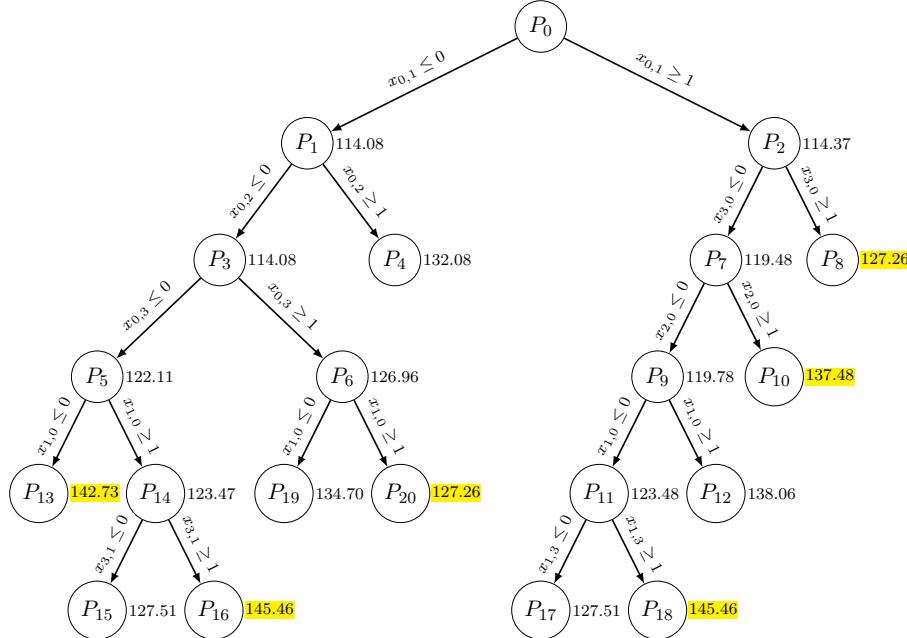
Heltalsproblemet opstilles som TSP1 mellem følgende fem punkter,

$$(10, 31), (1, 14), (43, 49), (23, 54), (22, 15).$$

Til at generere de nedre grænser bruges LP-relaksation som beskrevet tidligere, og vi relakserer derfor heltalskravet for  $x_{ij}$  i (5). Løsningen til dette problem har en objektfunktionsværdi på 114.08, som nu udgør en nedre grænse for den objektfunktionsværdi der findes i den optimale heltalsløsning. Branching foretages på en enkelt fraktionel variabel ad gangen. Eksempelvis er  $x_{01} = 0.25$  i den optimale løsning, og denne deles nu op i to delproblemer ved at tilføje hhv. begrænsninger  $x_{01} \leq 0$  og  $x_{01} \geq 1$ .

Til søgeræ er der brugt Best Bound Search (BBS), altså branches der videre ud af den gren, som har den mindste objektværdi. LP-problemet løses nu igen for hvert af de to delproblemer og processen gentages indtil der opnås en heltallig løsning eller indtil objektværdien ikke længere er

mindst. Branch and Bound søgetræet for dette eksempel kan ses i Figur 2.



Figur 2: Branch and bound træ, hvor heltalsløsninger er markerede med gul

I søgetræet er heltallige løsninger markeret med gult og  $P_i$  angiver rækkefølgen, der er branchet i. Objektværdien for løsningen til det optimale heltalsproblem er 127.26, da denne værdi opfylder heltalskrav for variablene og er den mindste nedre begrænsning for alle delproblemer i søgetræet. Som det kan ses i søgetræet findes der to løsninger med samme minimale objektfunktionsværdi. Det skyldes, at dette eksempel har symmetriske omkostninger, altså er  $c_{ij} = c_{ji}$  og den samlede omkostning er den samme uanset hvilken vej man går i kredsen. Pythonfilen `branch_and_bound_eksempel.py` viser hvordan træet er opnået.

## 2.4 CPLEX

Der findes mange forskellige algoritmer til løsning af lineær programmering (LP), heltalsprogrammering (IP) og en kombination af de to; Mixed-integer linear programming (MILP). Til LP-modeller er det typisk simplex algoritmen der bliver brugt, som også har givet navn til IBMs software CPLEX, som er en kommercial solver (IBM *What does CPLEX do?* 2019 [12]). CPLEXs MIP solver, benytter sig af blandt andet af branch-and-cut, som udover branch and bound gør brug af snitplansmetoden, men som vi ikke yderligere kommer ind på. Metoden er generel og standardindstillinger virker på forskellige MIP modeller, og det er da også IBMs anbefaling at bruge solveren direkte. (IBM *Invoking the optimizer for a MIP model.* 2019 [12]). Alternativt kunne man have brugt Gurobi, SCIP eller Googles GLOP. Dog blev CPLEX brugt i det førnævnte kursus og har været det foretrukne valg. Med hensyn til lange beregningstider, i forbindelse med at finde en optimal løsning, kan der sættes en relativ eller absolut *MIP gap tolerance*, som

stopper solveren når en brugbar løsningen er tæt nok på objektværdien for den bedste knude i søgetræet (IBM *relative MIP gap tolerance* 2019 [12]), (IBM *absolute MIP gap tolerance* 2019 [12]). Det vil altså sige at man accepterer en løsning der måske ikke er optimal. Alternativt, kan man sætte en begrænsning på beregningstiden.

# 3 Vehicle Routing Problem

Det er særligt relevant for ruteplanlægning i hjemmeplejen, at udvide modellen så der kan planlægges ruter for flere hjælpere samtidig. Vehicle Routing Problem er en fællesbetegnelse for ruteplanlægningsproblemer, hvor flere køretøjer skal besøge kunder, og den samlede omkostning minimeres. Omkostningen er typisk målt i tid eller afstand. Det blev introduceret af Dantzig og Ramser (1959 [8]) i 1959, i forbindelse med ruteplanlægning af lastbiler (Toth og Vigo 2001, kap. 1 [30]). Typisk starter køretøjerne i et depot, og skal slutte i samme depot - dog findes der varianter hvor køretøjet stopper ved kunden, eller andre depoter.

Som i afsnit 2; lad  $G = (V, A)$  være en graf, hvor  $V = \{0, \dots, n\}$  er punkter og  $A$  er sættet af kanter mellem punkterne i grafen. Depotet sættes i punkt 0. Derudover indføres notation for kunder,  $K = V \setminus \{0\}$ .  $C$  er igen en matrix, med omkostninger  $c_{ij}$  og  $c_{ii} = \infty$ . Beslutningsvariablene  $x_{ij}$  formuleres som før. Følgende VRP er opstillet med inspiration fra Toth og Vigo (2001, kap. 1 [30]).

**Optimeringsproblem med begrænsninger, VRP1:**

$$\text{minimer } \sum_{(i,j) \in A} c_{ij} x_{ij} \quad (8)$$

$$\text{med bibetingelser } \sum_{j \in V} x_{0j} = m \quad (9)$$

$$\sum_{i \in V} x_{i0} = m \quad (10)$$

$$\sum_{j \in V} x_{ij} = 1, \quad \text{for alle } i \in K \quad (11)$$

$$\sum_{i \in V} x_{ij} = 1, \quad \text{for alle } j \in K \quad (12)$$

$$+ \text{ subtur elimineringsbegrænsninger,} \quad (13)$$

$$x_{ij} \in \{0, 1\}, \quad \text{for } (i, j) \in A \quad (14)$$

Formuleringen minder om den forrige formulering af TSP men der kræves nu  $m$  afgange og ankomster fra depotet, som netop er hhv. betingelse (9) og (10). VRP kan således ses som en generalisering af TSP (Dantzig og Ramser 1959 [8]), hvorfor denne formulering også kaldes Multiple TSP (mTSP) (Bektas 2006 [4]).

Subturselimineringsbegrænsinger fra afsnit 4.1.1 og afsnit 4.1.2 kan stadig bruges.

## 3.1 Capacitated Vehicle Routing Problem

I en løsning til VRP1 ovenfor tages der dog, i sammenhæng med hjemmeplejen, ikke højde for antallet af borgere, hver hjælper skal besøge på en rute. Det vil sige, en hjælper kan få tildelt mange flere eller mange færre borgere end de andre. For at sætte begrænsning på antallet af

borgere der må besøges per hjælper, introduceres nu problemet 'Capacitated Vehicle Routing Problem' (CVRP). I CVRP ønskes der, ligesom i VRP, at finde en rute som minimerer en objektfunktion, men med den udvidelse, at hver  $i$ 'te kunde har en efterspørgsel på  $d_i$ , og at køretøjerne hver har samme kapacitet på  $q$ . CVRP består da i, at kundernes efterspørgsel på varer skal være opfyldt, samtidig med at kapacitetsbegrænsningerne på køretøjerne ikke er overskredet. Køretøjernes ruter kan starte og slutte i en eller flere depoter. Hvis antallet af køretøjer samt køretøjernes kapacitet ikke kan leve op til kundernes efterspørgsel, er der mulighed for at udelade en del af kunderne (Toth og Vigo 2001 [30]). Dog findes dette ikke særlig relevant i en kontekst af hjemmeplejen, da det kan gå ud over borgernes helbred, medmindre opgaven for eksempel er rengøring. Andre tilfælde af brug af CVRP modeller, kunne være ruteplanlægning for bus, postbud, lastbiler og skraldevogne (Toth og Vigo 2001 [30]).

### 3.1.1 Kapacitetsbegrænsninger

CVRP kan altså bruges til at begrænse antallet af borgere som hjælperne besøger,  $p$ , ved at sætte kapaciteten til  $p$ , samt efterspørgslen ved kunderne til 1. Det vil sige, en hjælper kan nu besøge  $p$  borgere før kapaciteten er opbrugt. Kapacitetsbegrænsningen til VRP1 kan da modellers som MTZ i (7). Kapacitetsbegrænsningerne her skal ses som både SEC og begrænsninger der sørger for at kapaciteten på et køretøj ikke er overskredet. Der præsenteres nu to forskellige kapacitetsbegrænsninger, som sættes ind på (13) i VRP1.

**3.1.1.1 MTZ** Det kan være brugbart at modellere en generalisering af begrænsning (7), hvis kapaciteten afhænger af efterspørgslen på en vare og ikke blot er antallet af besøg, og køretøjerne kun kan have et bestemt antal varer. Dette kan være relevant i hjemmeplejen, hvis en hjælper eksempelvis skal ud at handle for en borger eller have særligt udstyr med på ruten. Her kan man skelne mellem om en hjælper har kapacitet til dette eller ej ved at tilføje kapacitetsbegrænsninger på køretøjerne. Det antages her, at hvert køretøj har samme kapacitet  $q$ . Så er

$$u_i - u_j + qx_{ij} \leq q - d_j, \quad \text{for alle } i, j \in K, i \neq j, d_i + d_j \leq q \quad (15)$$

$$d_i \leq u_i \leq q, \quad \text{for alle } i \in K \quad (16)$$

en begrænsning der opfylder kravene i afsnit 3.1. Der introduceres  $u_i$  variable for hver kunde, hvor  $q$  er kapaciteten og  $d_j$  er efterspørgslen på varer (Desrochers et al. 1991 [9]). Bibetingelse (15) udelukker ruter hvor, efterspørgslen overstiger køretøjets kapacitet, samtidig med at det udelukker subture. Det ses også at hvis man som nævnt sætter  $q = p$  og  $d_i = 1$  for alle  $i \in K$ , så fås netop (7). Formuleringen VRP1 med (15) og (16) kaldes her CVRP\_MTZ.

**3.1.1.2 Generalized subtour elimination constraints (GSEC)** Alternativt kan kapacitetsbegrænsningerne modelleres med *Generalized subtour elimination constraints* (GSEC) (Toth

og Vigo 2001, kap. 3 [30]). En subtur  $S$  er en kreds med  $|S| < n$  punkter, som ikke indeholder depotet. Altså udelukker

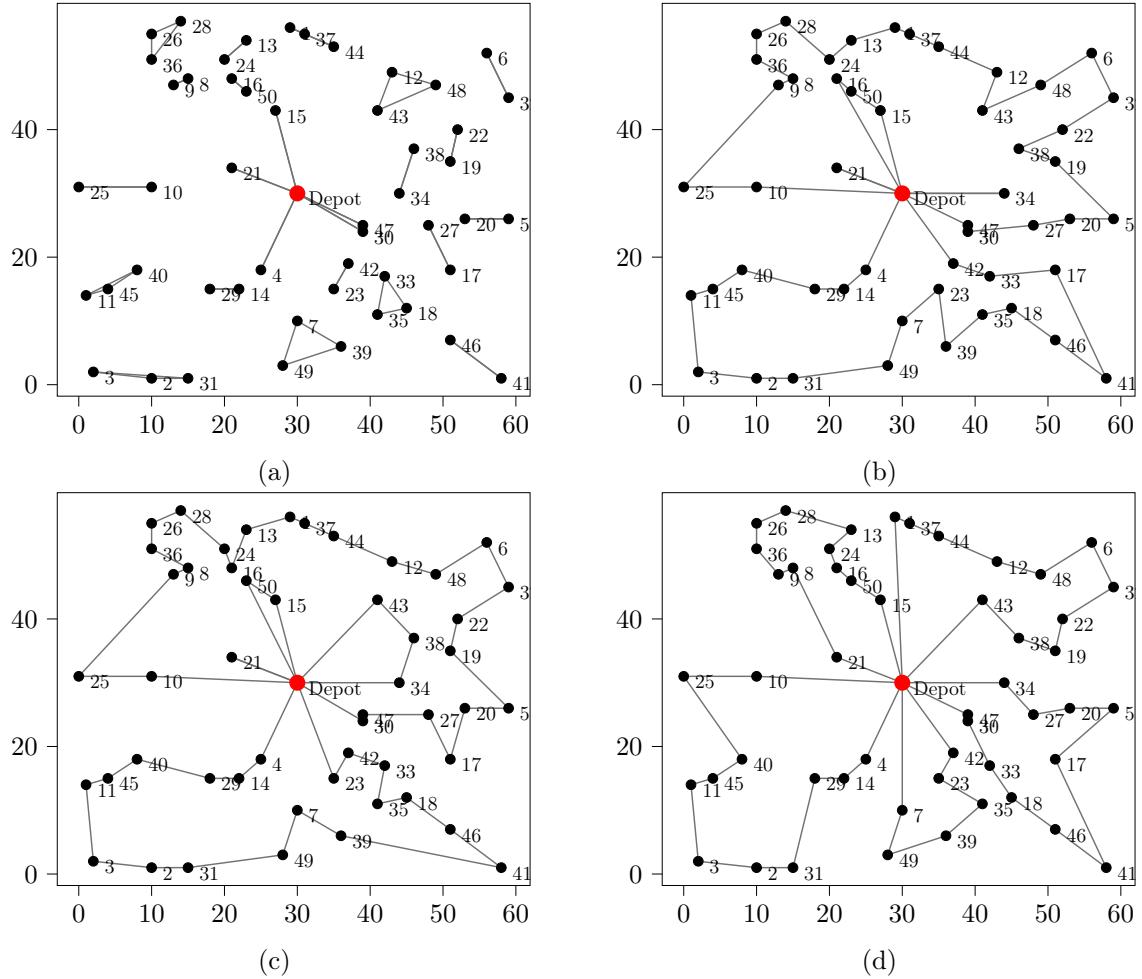
$$\sum_{i,j \in S} x_{ij} \leq |S| - k(S), \quad \text{for } S \subseteq K, |S| \geq 1 \quad (17)$$

subture der overstiger kapacitet på et køretøj. Dette kan fortolkes som, at der i delmængden  $S$  skal gå mindst  $k(S)$  kanter ud af  $S$ . Altså kræves der mindst  $k(S)$  vogne til at besøge punktmængden  $S$ . Hvis  $k(S) = \lceil \sum_{j \in S} d_j/q \rceil$  i (17) kaldes det da Rounded Capacity Constraints (RCC). Til sidst bemærkes det, at en kant  $(i, j)$  kan udelades fra formuleringen hvis det direkte overskridt kapaciteten  $q$ . Det vil sige hvis  $d_i + d_j > q$ , som nævnt i Kallehauge og som også er brugt i (15) (Kallehauge 2008 [14]). Formuleringen VRP1 med (17) kaldes her CVRP\_DFJ.

Et yderligere alternativ til dette er en formulering af Gavish and Graves, som omtalt i Letchford et al. (2006 [17]), hvor man introducerer en ny variabel  $f_{ij}$ , som er den samlede mængde, der er blevet leveret af bilen til og med punkt  $i$  på ruten, såfremt  $x_{ij} = 1$ . Herefter benyttes denne til at sørge for, at mængden der er blevet leveret på ruten når køretøjet forlader punkt  $i$  er lig med den mængde, der var leveret lige inden punkt  $i$  plus efterspørgslen ved punkt  $i$ . Derudover har  $f_{ij}$ -variablene en øvre grænse på bilens maksimum kapacitet. Dette sikrer nu, at bilerne ikke leverer mere på deres rute, end der er kapacitet til.

### 3.1.2 Anvendelse af CVRP

For at illustrere brug af CVRP, ses i Figur 3 en optimalløsning fra en implementering af CVRP\_DFJ, hvor kapaciteten er  $q = 11$  og antallet af hjælpere er  $m = 5$ . Efterspørgslen er sat til 1 for alle borgere.



Figur 3: Iteration 1 (a), 30 (b), 60 (c) og 115(d) ud af 115 for CVRP med DFJ-SEC

Dette bliver det brugt til at begrænse antallet af borgere der bliver besøgt, således at en hjælper ikke besøger for mange borgere. Dels fordi der ikke er fundet data på hvad en hjælper har kapacitet til at have med i bilen og heller ikke data på, hvilke ting der skal med bilen. Dels for at imødekomme ønsket om at en hjælper knytter tættere familiære forhold til borgeren, hvilket antages at være sværere jo flere der besøges, samt for at balancere arbejdsbyrden indbyrdes mellem hjælpere.

Kapacitetsbegrænsningerne med en efterspørgsel på en, sikrer en øvre grænse for længden af en rute, det vil sige, lange ruter udelukkes. Der da kan forekomme ruter med få besøg, afhængigt af kapacitet, antal borgere og omkostninger, og fordi der ikke betragtes en nedre grænse. Det er altså ikke nødvendigvis nok at begrænse antallet af besøg, da opgaverne kan variere i både længde og sværhedsgrad, og enkelte hjælpere kan da have en køreliste der består udelukkende af nemme og hurtige opgaver, eller lange tunge opgaver. Alligevel kan man på én måde, ved hjælp af fastsættelse af antal ruter, sørge for at ruterne er nogenlunde samme længde. Da der er 50 borgere, kan den mindst mulige rute være på 6 borgere, hvis det er optimalt. Til løsning af CVRP\_DFJ bruges nogenlunde samme fremgangsmåde som i afsnit 2.2. VRP uden RCC ses i

første iteration, som frembringer løsningen der ses i figur 3a. Dernæst findes alle cykler og RCC (17) tilføjes til de cykler, der ikke indeholder depotet, altså subture. Hvis en cykel  $C$  indeholder depotet og antallet af borgere i cyklen  $|C| - 1$  er større end  $q$ , så bryder  $S = C \setminus \{0\}$  begrænsningen (17) og derfor tilføjes der også RCC for disse ture. Programmet stoppes igen, såfremt der ikke er forskel på cyklerne i to iterationer. CVRP\_MTZ løses direkte med en tidsbegrænsning på 30 minutter, da denne tog lang tid at løse til bevist optimalitet, med samme parametre  $q = 11$  og  $m = 5$ . Begge formuleringer gav samme resultatet i Figur 3d. Det tog 115 iterationer for CVRP\_DFJ implementering at finde frem til løsningen. Objektværdi og køretid kan ses i afsnit 4.1. I den optimale løsning en kort tur med kunderne  $(0, 7, 49, 39, 35, 23, 42)$ , som kun besøger 6 kunder, som måske ikke er hensigtsmæssigt, men  $q$  er valgt på baggrund af artiklen (Røge 2017 [28]), som nævnt i dataafsnittet 1.31. Pythonkoden findes i CVRP\_DFJ.py og CVRP\_MTZ.py.

## 3.2 Capacitated Vehicle Routing Problem med Tids-vinduer

I hjemmeplejen sættes der stor pris på, at hjælperne kommer på nogenlunde faste tidspunkter hver gang. Det skaber forudsigtighed og tryghed hos borgerne. Fastlagte besøgstidspunkter eller tidsintervaller hvor borgeren skal besøges, kan modelleres ved hjælp af tidsvinduer. Hvis der ovenikøbet er kapacitetsbegrænsninger, kaldes dette i litteraturen for 'Capacitated Vehicle Routing Problem with Time Windows' (CVRPTW), eller blot VRPTW Toth og Vigo (2001, afsnit 2.1.1[30]). Oftest modelleres der et tidsvindue  $[a_i, b_i]$  for kunde  $i$ , hvori serviceringen af kunde  $i$  skal begynde. Derudover er der en varighed  $p_i$  på, hvor lang tid servicen tager for hver kunde. Det er muligt at ankomme til en kunde inden servicering, men i det tilfælde ventes der med start af servicering til det tidligst mulige (Kallehauge 2008 [14]).

### 3.2.1 MTZ

For at overholde tidsvinduerne er det nødvendigt at have andre begrænsninger end vist tidligere. En MTZ-lignende begrænsning som den følgende kan bruges i VRP1

$$s_i - s_j + (b_i + t_{ij} - a_j) x_{ij} \leq b_i - a_j, \quad \text{for } i, j \in K \quad (18)$$

Denne sørger for, at borgerne besøges inden for tidsvinduerne. Variablen  $t_{ij}$  er tiden fra man ankommer til  $i$  til man ankommer til  $j$  - det vil sige  $t_{ij} = p_i + c_{ij}$ . Altså den tid det tager fra kunde  $i$  til kunde  $j$  inklusiv hvor længe kunde  $i$  skulle serviceres. Der tilføjes altså endnu flere variable  $s_i$  til modellen, hvor  $s_i$  fungerer som variable der holder styr på hvornår servicering starter for kunde  $i$ , som vi her kalder serviceringstidspunkt. Derudover tilføjes begrænsningen:

$$a_i \leq s_i \leq b_i \quad \text{for alle } i \in K \quad (19)$$

(19) sørger for at serviceringenstidspunktet ligger inden for tidsvinduet. Kombination af (15), (16), (18) og (19) blev foreslæbt af Bard et al. (2002 [3]) og kaldes her **CVRPTW1**. I artiklen minimeres dog antallet af køretøjer, for at dække alle besøg ved at minimere outflow fra depotet. Notation af begrænsninger er her taget fra Kallehauge (2008 [14]).

### 3.2.2 Infeasible Path Elimination Constraints

En anden måde at overholde tidsvinduerne på er, at udelukke de stier som ikke er brugbare. Det vil sige stier der bryder tidsvinduerne. Hvis der er tale om CVRP, er en sti heller ikke brugbar hvis stiens samlede efterspørgsel overstiger  $q$ . Begrænsninger der udelukker stier der ikke overholder tidsvinduer kaldes Infeasible Path Elimination Constraints i artiklen af Ascheur et. al. (2001 [2]), som foreslog de såkaldte tournament constraints. Lad  $P = (v_1, v_2, \dots, v_k)$ , være en sti hvor  $k$  er antallet af kunderne. Så kan

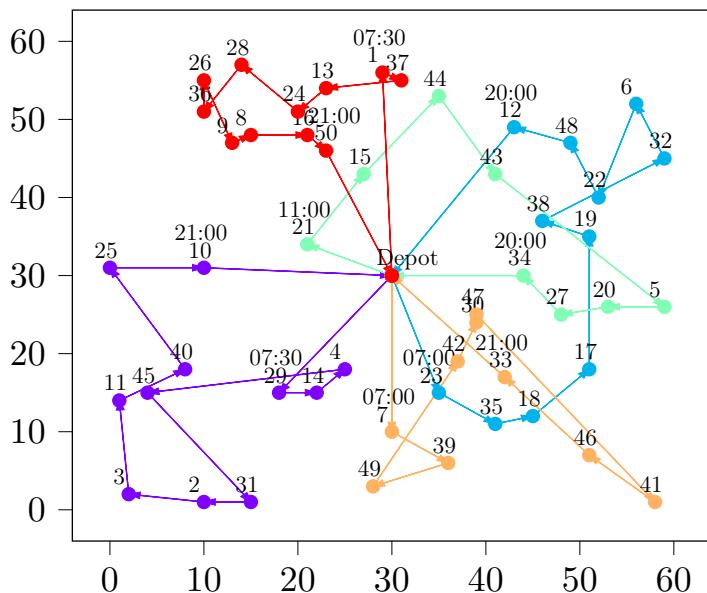
$$\sum_{i=1}^{k-1} \sum_{j=i+1}^k x_{v_i v_j} \leq k - 2 \quad (= |P| - 1), \quad \text{for alle ikke-brugbare stier } P \quad (20)$$

bruges til at løse VRPTW. (20) umuliggør alle ikke-brugbare stier ved at begrænse antallet af kanter i stien. For at finde ud af om en sti er ikke-brugbar kan serviceringstidspunkterne for kunderne findes. Givet stien  $P$ , så starter serviceringen tidligst hos den første kunde ved  $s_{v_1} = a_{v_1}$ . Start af servicering hos de efterfølgende kunder, er da givet ved:

$$s_{vi} = \max(s_{v_{i-1}} + t_{v_{i-1}v_{v_i}}, a_{v_i}) \quad (21)$$

hvor  $t_{ij}$  og  $s_i$  er de samme som i afsnit 3.2.1. Såfremt  $s_{v_i} > b_{v_i}$  eller at den samlede efterspørgsel på kunderne i  $P$  er større end  $q$  er stien ikke-brugbar (Kallehauge 2008 [14]). En model med VRP1, RCC (17) og tournament constraints (20) kaldes her **CVRPTW2**.

### 3.2.3 Anvendelse af CVRPTW1



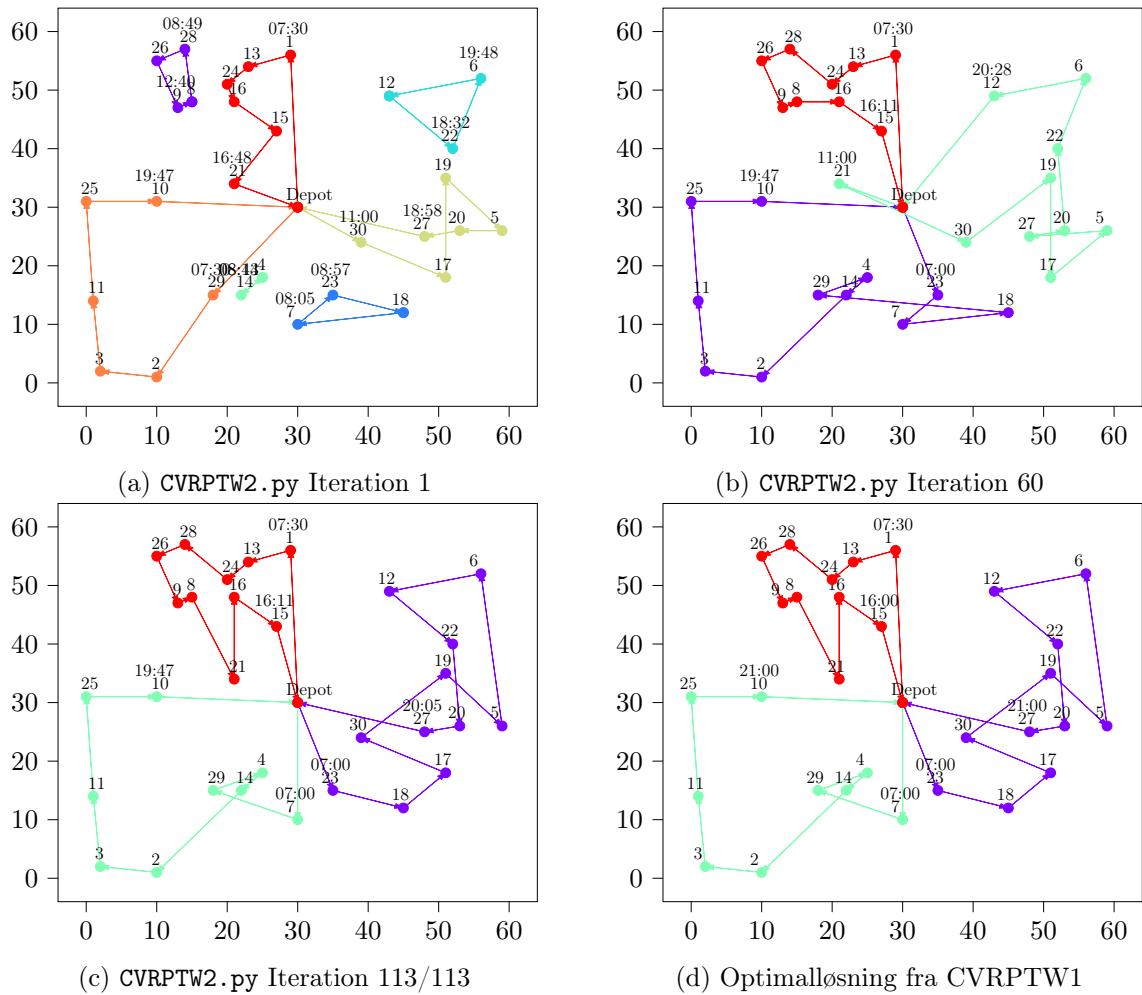
Figur 4: VRPTW på datasæt

Figur 4 viser en orienteret graf som er optimalløsningen til formuleringen **CVRPTW1**, på *Borgerdata*, hvor hver hjælper maksimalt må besøge  $q = 11$  borgere og hvor depotet har  $m = 5$  hjælpere. Hver hjælpers rute har fået en farve, og af hensyn til figurens overskuelighed er ankomsttidspunktet kun vist for den første og sidste borger i hver rute. Grafen er vist orienteret, grundet kravet om tidsvinduer. Det ses at nogle ruter er nødt til at krydse over, hvor afstanden ikke nødvendigvis er minimal, men det er påkrævet grundet kravet om at lave besøg indenfor tidsvinduerne. Formuleringen muliggør altså at borgerne besøges indenfor de korrekte tidsvinduer, samt at hjælperne besøger et begrænset antal borgere samtidig med at den samlede afstand minimeres. Imidlertid kan det give anledning til lange vagter, for eksempel ruten markeret med rød i Figur 4, hvor hjælperen nødvendigvis kører ud af depotet før kl. 8 og vender tilbage senere end kl. 21, hvilket ikke er hensigtsmæssigt, hvis der skal tages højde for arbejdstider. Med det meneres der at det antages, at der grundet overenskomster ikke må være hjemmehjælpere der arbejder så længe. Arbejdstider kan for eksempel imødekommes ved at dele datasættet op i borgere der skal besøges mellem kl 7 og kl 14 og de borgere der skal besøges efter kl 14, således, at der løses et problem for, hvad der her kaldes et morgenhold og et aftenhold. I hver af de to delproblemer, kan der eventuelt sættes et tidsvindue på depotet, for at sørge for at hjælperne når tilbage inden vagtskifte. I så fald kan der laves en kopi af depotet i punkt  $n + 1$ , da  $s_i$  variablene i **CVRPTW1**, kun er defineret for punkter forskellig fra depotet. Det kan for eksempel bruges hvis der er et begrænset antal køretøjer, som skal deles mellem morgenholdet og aftenholdet, og morgenholdet skal nå tilbage til depotet inden et bestemt tidspunkt. Derudover kan man overveje at sætte en øvre grænse på varigheden af en rute. Dog vil den tilføjelse ikke garantere gængse morgen og

aftenvagter, da den for eksempel ikke vil udelukke en rute fra kl 12-20, hvis den øvre grænse er på 8 timer.

### 3.2.4 Anvendelse af CVRPTW2

I implementeringen af CVRPTW2 løses VRPet fra afsnit 3, dernæst løbes der gennem alle cykler i løsningen, og der tilføjes RCC som i afsnit 3.1.2. Derudover tjekkes der for Infeasible Path Constraints, ved at beregne de tidligste serviceringstidspunkter, som beskrevet i afsnit 3.2.3. Af hensyn til implementering tjekkes serviceringstidspunkter kun for kunder, i de cykler der indeholder depotet, da cykler der ikke indeholder depotet bliver udelukket af RCC. Depotet har en sen lukketid, så man kan altid nå tilbage til depotet. Derfor er det korrekt at tjekke at cykler indeholdt depotet, selvom disse ikke er stier. For hver kunde  $i$  der bliver besøgt i cyklen, tjekkes om  $s_i > b_i$ . Hvis serviceringstidspunkt er senere end slutvinduet så er stien til og med kunde  $i$  ikke-brugbar og der tilføjes (20) uden depotet. Igang stoppes programmet når der i to iterationer i streg er nøjagtig samme løsning, som så vil være brugbar og optimal.



Figur 5: Iterationer af CVRPTW2 og sammenligning med CVRPTW1

Der blev forsøgt at løse samme problem som i 3.2.1, men efter 30 minutters kørsel var der ingen brugbar og optimal løsning. I stedet vises et eksempel i Figur 5, heltalsløsninger der fremkommer på de første 30 borgere i Borgerdata, med  $q = 11$  og  $m = 3$ . I Figur 5c og Figur 5d, er henholdsvis løsningen for CVRPTW2 og CVRPTW1. En bemærkelsesværdig observation er, at nogle af serviceringstidspunkterne er forskellige. For eksempel serviceres borger 10 på forskellige tidspunkter. Seneste afgang fra borger 25 er kl 20:00 med serviceringstid på 00:10, men servicering af borger 10 er kl 21:00 i Figur 5d, selvom borgerne er relativ tæt på hinanden. Derfor kan man når ruterne er fundet i model CVRPTW1, justere serviceringstidspunkterne med proceduren ved (21) (Kallehauge 2008, afsnit 4 [14]). Pythonkoden til plots og løsning findes i CVRPTW2.py og CVRPTW1.py, hvor der kun indlæses, nrows = 31 datapunkter.

**3.2.4.1 Morgenhold og aftenhold** For at vise hvordan man kan imødekomme arbejdstiderne vises et eksempel på hvordan, det kan gøres ved at dele data op som nævnt. Da CVRPTW2, var langsommere end CVRPTW1 tages der derfor udgangspunkt i denne. Der skal foretages et valg om, hvordan datasættet partitioneres, og her vælges der at dele borgerne op hvor tidsvinduet slutter senest kl. 14:00. Dog vil der kunne forekomme overlap, forstået på den måde at et tidsvindue kan starte før kl. 14 og slutte efter kl 14, for eksempel borger 9, hvis startvindue er kl. 12 og slutvindue er kl 16 (se bilag 1). Da der i modellen ikke er angivet en servicering i depotet, og dermed afgangstidspunkt, kan man risikere at punkt 9 serviceres kl 12 af aftenholdet i en optimalløsning, selvom det ønskes at sådan et besøg skulle dækkes af morgenholdet. Derfor kan der for aftenholdet tilføjes begrænsninger på  $s_i$  variablene, hvor serviceringstidspunktet for kunde  $i$  skal være kl 14:00 eller senere. Det er nu uklart hvordan, de fem hjælpere skal fordeles på et morgenhold og et aftenhold, og om det i det hele taget er nok med fem hjælpere. For at finde det mindste antal hjælpere, der maksimalt må besøge  $q = 11$  borgere, kan objektfunktionen (8) i CVRPTW1, erstattes med

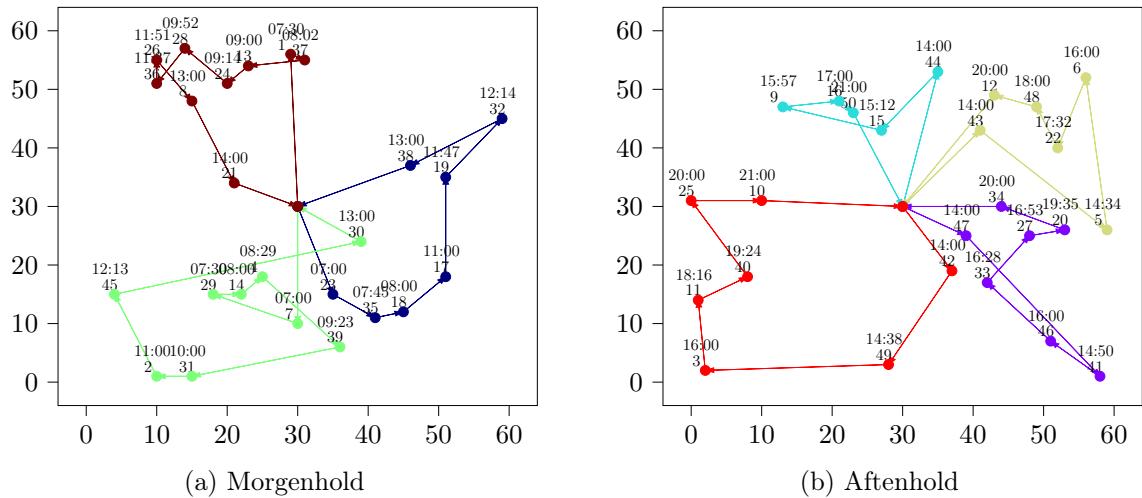
$$\text{minimer } \sum_{i \in K} x_{0i} \quad (22)$$

som netop er antallet af kanter ud af depotet, og derfor antallet af hjælpere. Derudover erstattes betingelserne (9) - (12) med

$$\sum_{j \in V} x_{ij} = 1, \quad \text{for alle } i \in K \quad (23)$$

$$\sum_{j \in V} x_{ij} - \sum_{j \in V} x_{ji} = 0, \quad \text{for alle } i \in V \quad (24)$$

hvor (23) og (24) sikrer at hver borger bliver besøgt og forladt af et køretøj, men nu uafhængigt af antallet af køretøjer. Formuleringen er taget fra Bard et al. (2002 [3]) I praksis kan man løse CVRPTW1 med en solver, og starte med at se om solveren kan løse problemet med  $m = 1$ , og derefter forøge  $m$  med 1 såfremt solveren ikke giver nogen løsning.



Figur 6: CVRPTW fordelt på morgen- og aftenhold

Opdelingen som beskrevet før giver tilfældigvis 25 borgere på hvert hold. Det mindste antal hjælpere til at servicere morgenholdet blev  $m = 3$ , og for aftenholdet blev det  $m = 4$ . Dernæst blev CVRPTW1 løst på hvert hold med det mindste antal hjælpere, samt  $q = 11$ . De optimale løsninger ses på Figur 6, og implementeringen findes i mapperne **Morgen** og **Aften** og har begge filnavn **CVRPTW1\_hold.py**.

### 3.2.5 Betydning af tidsvinduerne s længde

Besøgs-/tidsvinduerne spænder, som før nævnt, mellem 30 minutter og 5 timer. Det vides ikke, om disse meget forskellige længder på tidsvinduer er repræsentative for virkeligheden, men der vil forsøges at undersøge og overveje betydningen af tidsvinduernes længde. Løsningsrummet er stort hvis tidsvinduerne er for store og det kan derfor tage lang tid at finde en løsning. Derudover har flere ældre ved hjemmeplejen givet udtryk for, at det er værdsat med så faste besøgstidspunkter som muligt, samt at store intervaller er irritationsmomenter (PwC side 21. [24]). Bliver tidsvinduerne så små, at de bliver et ankomsttidspunkt, kollapsed problemet til et såkaldt Vehicle Scheduling Problem (VSP) (Raff 1983 [25]). Et VSP tager en række opgaver, som hver indeholder starttidspunkt, sluttidspunkt og start- og slutlokation og forsøger at minimere en objektfunktion. I forbindelse med hjemmeplejen, ville depotet både være start- og slutlokation, og objektfunktionen der skulle minimeres er kørselstid (eller afstand). Her kunne man også minimere antallet af ruter og dermed antallet af biler og samlet køretid. Et VSP er hurtigt at løse, da antallet af mulige løsninger er begrænset væsentligt. Det er derfor generelt foretrukket at holde tidsvinduerne små, både for borgeren og for løsningstiden. Hvis det tager for lang tid at løse et problem kan man reducere tidsvinduerne hvis trekantsuligheden for kørselstiderne er overholdt (Kallehauge et. al. 2005, afsnit 7 [15]).

Den gennemsnitlige længde for besøgsvinduerne ved de 50 borgere i datasættet i Tabel 1 er 2 timer og 36 minutter. For at undersøge forskellen i kørselstider ved forskellige længder af besøgsvinduer,

er problemet i afsnit 3.2.3 løst yderligere to gange på det samme datasæt, men med gennemsnitlige besøgsvinduer på hhv. 50 minutter og 4 timer og 35 minutter, hvor tidsvinduerne er gjort mindre og bredere manuelt, i henholdsvis Borgerdata\_smal og Borgerdata\_bred.

Model	Gennemsnitlig tidsvindue	Kørselstid i sekunder	Objektværdi	Data
CVRPTW1	02:36	0.61	605.63	Borgerdata
CVRPTW1	00:50	0.5	825	Borgerdatasmal
CVRPTW1	04:35	107.44	544.16	Borgerdatabred

Tabel 2: Tidsvinduer

### 3.2.6 Tre-indeksering

I hjemmeplejen er det relevant at kunne skelne mellem hjælperne når der laves ruter. Eksempelvis kan der være forskel på hjælpernes kompetencer samt hvilket slags køretøj hver enkelt hjælper har til rådighed. Det har en indflydelse på hvem hjælperen kan og skal besøge. Derudover kan hjælperne have forskellige arbejdstider. Det er derfor relevant at introducere formuleringer, der benytter sig af tre-indeksering. Hidtil er der præsenteret såkaldte to-flows-indekseringsmodeller, altså de  $x_{ij}$  variable, som indikerer om en kant fra kunde  $i$  til kunde  $j$  benyttes. I tre-indekseringsformuleringer formuleres beslutningsvariable ofte som  $x_{ijk}$ , hvor  $x_{ijk} = 1$  hvis køretøj  $k$  benytter kanten fra kunde  $i$  til kunde  $j$ . Her præsenteres en tre-indeks formulering som er tungt inspireret af Kallehauge et. al. (2005, kapitel 3 [15]). Lad  $H$  være et sæt af køretøjer,  $K = 1, \dots, n$  et sæt af kunder, og  $G = (V, A)$  en orienteret graf, hvor  $V = K \cup \{0, n+1\}$ , hvor depotet er indekseret i 0, samt  $n+1$ . Der er defineret kanter  $(i, j)$ , hvor  $i! = j, i! = n+1, j! = 0$ . Hver kant  $(i, j)$  har en omkostning  $c_{ij}$ , og en tid  $t_{ij} = c_{ij} + p_i$  som før. Hvert køretøj har en kapacitet på  $q_k$  og hver kunde har en efterspørgsel på  $d_i$ . Tidsvinduer er igen givet ved  $[a_i, b_i]$  for  $i \in V$  (Kallehauge 2005 [15]).

### Optimeringsproblem med begrænsninger, CVRPTW3:

$$\text{minimer } \sum_{k \in H} \sum_{(i,j) \in A} c_{ij} x_{ijk} \quad (25)$$

$$\text{med bibetingelser } \sum_{k \in H} \sum_{j \in V} x_{ijk} = 1, \quad \text{for alle } i \in K, \quad (26)$$

$$\sum_{i \in K} d_i \sum_{j \in V} x_{ijk} \leq q_k, \quad \text{for alle } k \in H, \quad (27)$$

$$\sum_{j \in V} x_{0jk} = 1, \quad \text{for alle } k \in H, \quad (28)$$

$$\sum_{i \in V} x_{ihk} - \sum_{j \in V} x_{hjk} = 0, \quad \text{for alle } h \in K, \text{ for alle } k \in H, \quad (29)$$

$$\sum_{i \in V} x_{i,n+1,k} = 1, \quad \text{for alle } k \in H, \quad (30)$$

$$s_{ik} + t_{ij} - M_{ij} (1 - x_{ijk}) \leq s_{jk}, \quad \text{for alle } i, j \in V, \text{ for alle } k \in H, \quad (31)$$

$$a_i \leq s_{ik} \leq b_i, \quad \text{for alle } i \in V, \text{ for alle } k \in H, \quad (32)$$

$$x_{ijk} \in \{0, 1\}, \quad \text{for alle } i, j \in V, \text{ for alle } k \in H. \quad (33)$$

Objektfunktionen (25) minimerer igen den totale afstand. Betingelse (26) sørger for at kunder besøges en gang. (27) sørger for at hver vogn ikke overstiger kapaciteten på vognen. Betingelse (28) sikrer at alle vogne forlader depotet. Betingelse (29) sørger for at når et køretøj besøger en kunde, forlader køretøjet også kunden igen. Betingelse (30) sørger for at køretøjet ender i depotet  $n+1$ . Betingelse (31) minder om MTZ i CVRP og sørger for at serviceringstidspunkterne mellem punkterne. Betingelse (32) sørger for serviceringen er indenfor tidsvinduerne.  $s_{ik}$  variablene er irrelevante hvis køretøj  $k$  ikke besøger kunde  $i$ . Den sidste betingelse er heltalskrav.  $M_{ij}$  kan erstattes af  $\max(b_i + t_{ij} - a_j), (i, j) \in A$  (Kallehauge 2005 [15]).

### 3.2.7 Synkronisering

I litteraturen er synkronisering en af de seneste udviklinger af VRP. Med synkronisering menes der, at køretøjernes køreplaner er afhængige af hinanden. En oversigtsartikel af synkronisering findes i Drexl (2012 [11]). I en hjemmeplejekontekst, kan synkronisering være nyttig at modellere. Dette kan for eksempel være hvis man skal være to om at løfte eller bade en borger, eller hvis medicin skal gives et stykke tid efter et måltid, som Bredström og Rönnqvist (2008 [5]) nævner i deres artikel. Synkronisering kan også anvendes i andre kontekster som for eksempel sikkerhedsvagter eller politi, som skal være på bestemte lokationer og hvor nogle af opgaverne kræver to eller flere samtidig. Et andet eksempel kunne være træfældning, hvor træet skal hentes efter en bestemt periode efter fældning, som også er nævnt i artiklen. Konkret til modellering af synkronisering kommer Dohn et. al. (2011 [10]) med følgende forslag til omskrivning af (32)

$$a_i \sum_{j \in V} x_{ijk} \leq s_{ik} \leq b_i \sum_{j \in V} x_{ijk} \quad \text{for alle } i \in K, \text{ for alle } k \in H \quad (34)$$

Dette sørger for at  $s_{ik} = 0$  når køretøj  $k$  ikke besøger kunde  $i$ . Hvorimod (32) blot tvang  $s_{ik}$  til at ligge i tidsinduet, uanset om køretøj  $k$  besøger kunde  $i$ . Da disse nu kun antager en værdi større end 0, når køretøjet rent faktisk besøger kunde  $i$ , introduceres  $\delta_{ij}$ , som er den mindste forskel i ankomsttidspunkt fra besøg af kunde  $i$  til kunde  $j$ . Ved brug af deres notation lades  $\Delta$  være det sæt af punkter  $(i, j)$  hvor der findes en tidsmæssig afhængighed. Så er *generalized precedence constraints* givet ved:

$$\sum_{k \in H} s_{ik} + \delta_{ij} \leq \sum_{k \in H} s_{jk} \quad \text{for alle } (i, j) \in \Delta \quad (35)$$

hvor  $\sum_{k \in H} s_{ik}$  under (34), er serviceringstidspunkt for besøg  $i$ .

Betingelse (35) kan modellere tidsafhængighed mellem to besøg af forskellige punkter, altså ikke hvis ét punkt skulle besøges på samme tid. Dette skyldes at (35) altid vil være opfyldt hvis  $i = j$  samt  $\delta_{ij} = \delta_{ji} = 0$ , hvor sidstnævnte er påkrævet for synkronisering, og hvor man desuden har ændret i (26), for at lade en kunde blive besøgt to gange. Da uligheden altid vil være opfyldt, kan der med (35) ikke garanteres for at besøgene sker på samme tid. En løsning på det vil være at man kopierer de punkter der kræver synkronisering og sørge for synkronisering mellem de oprindelige punkter samt deres kopi.

### 3.2.8 Anvendelse af tre-indeksering

Med tre-indeksering er der mulighed for at sætte begrænsning på hver vogn. For at modellere arbejdstider tilføjes begrænsningerne

$$\begin{aligned} s_{0k} &\geq \text{vagtstart}_k \\ s_{(n+1)k} &\leq \text{vagtslut}_k \end{aligned} \quad (36)$$

Inspirationen til disse begrænsninger er fra Rasmussen et al. (2012 [27]), som også modellerer en arbejdstid for hjemmehjælpere. I deres artikel maksimeres også borgernes præferencer for en bestemt hjælper ved at have en omkostning for en kombination af hjælpere og borgere. Disse begrænsninger (36) påbyder hver  $k$ 'te hjælper at forlade depotet 0 til tidligst vagtstart $_k$  og vende tilbage til depotet inden vagtslut $_k$ . Sammen med kapacitetsbegrænsningen (26), fås da den ønskede model, der både kan begrænse antallet af besøgte på en vagt og differenciere mellem hjælperne. CVRPTW3 med betingelse (34) i stedet for (32), samt (35) og (36) kaldes **CVRPTWTD1**.

Der vil nu forsøges at løse CVRPTWTD1 *Borgerdata*. I det følgende eksempel, simuleres et scenarie hvor en plejer er blevet syg og to vikarer er kaldt ind. Vikarerne besøger færre end

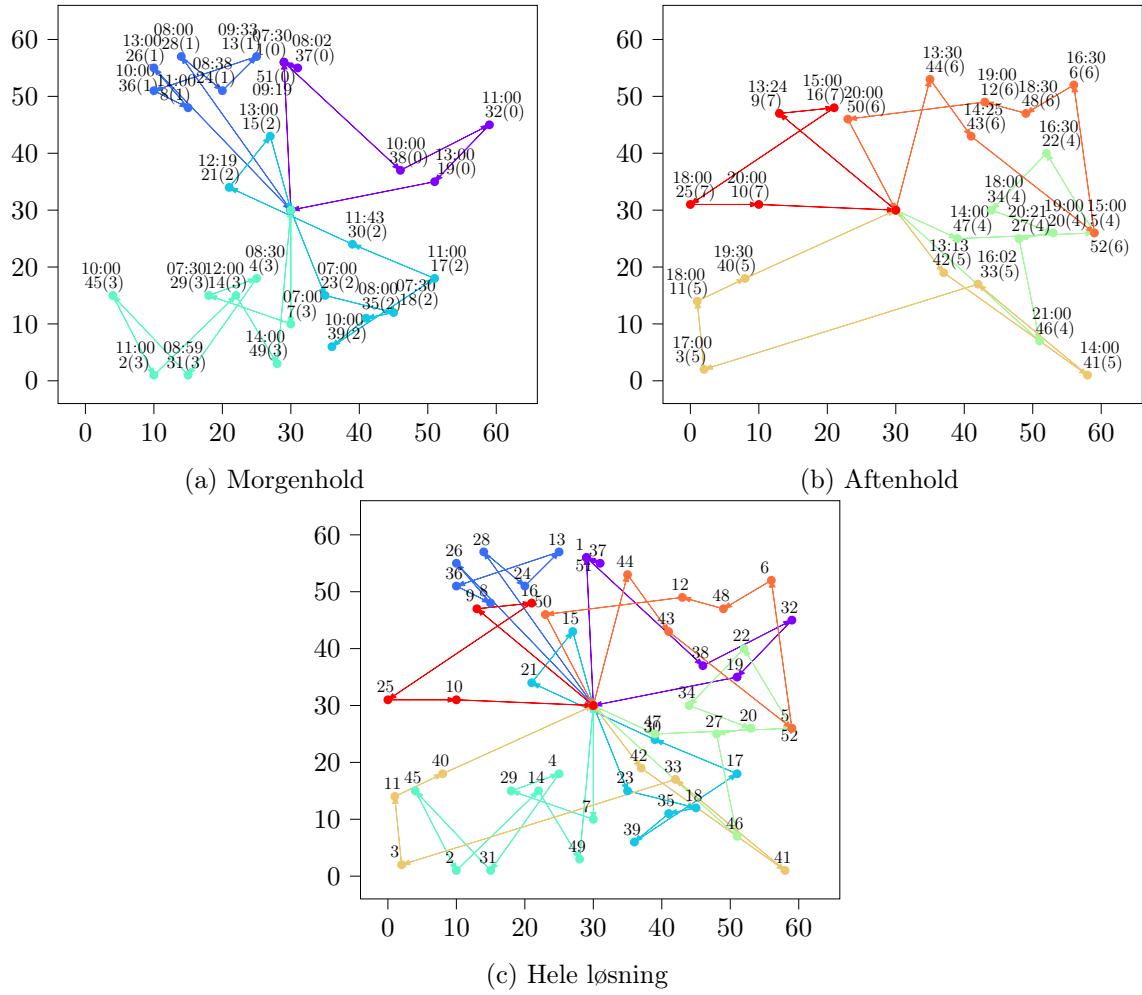
de faste hjælpere. Af løsningen fra afsnit 3.2.3.1 antages at der skal bruges 3 hjælpere på et nyt morgenhold og 4 på et nyt aftenhold. Holdene er nye, da datasættet i afsnit 3.2.3.1, blot blev delt op. Det havde den konsekvens at ankomsttidspunktet tilbage til depotet ikke kunne garanteres. Man kunne have delt datasættet op anderledes, så hvis hjælperne på morgenholdet skulle være tilbage kl 14:00 kunne de for eksempel senest servicere en borger kl 13:30. Men nu er der altså mulighed for eksplisit at sætte et slutvindue på depotet for den enkelte hjælper, hvilket kan garantere at vagten slutter på et bestemt tidspunkt, hvilket giver anledning til en anden opdeling af borgerne end i afsnit 3.2.3.1. En fast medarbejder på det nye morgenhold er blevet syg og to vikarer kaldes ind. Da kapaciteten tidligere ikke betød noget i afsnit 3.2.3.1, prøves med  $q = 8$  for de fastansatte og  $q = 6$  for de to vikarer. Der antages at borger 5 kræver 2 fastansatte på samme tid, for eksempel tungt løft og bad, under hele besøget. Derudover antages at borger 1 får morgenmad ved første besøg, og skal have medicin tidligst 1.5 time efter første besøg, og ikke senere end 3 timer efter. Der laves derfor en kopi af borger 5 med samme adresse, besøgsvindue og servicetid, en kopi af borger 1 med besøgsvindue 9:00-11:00 og servicetid på 00:15 til at give medicin. Disse sættes som nye borgere i datasættet. Tabel 3 giver en oversigt over hjælpere og Tabel 4 viser slutningen af det nye datasæt brugt på CVRPTWTD1.

$k$	$q_k$	Vagtstart $_k$	Vagtslut $_k$	Hold	Status
0	6	6:00	15:00	Morgen	Vikar
1	6	6:00	15:00	Morgen	Vikar
2	8	6:00	15:00	Morgen	Fastansat
3	8	6:00	15:00	Morgen	Fastansat
4	8	13:00	22:00	Aften	Fastansat
5	8	13:00	22:00	Aften	Fastansat
6	8	13:00	22:00	Aften	Fastansat
7	8	13:00	22:00	Aften	Fastansat

Tabel 3: Oversigt over hjælpere

Navn	Adresse	Besøgsvindue	Servicetid
51(1)	(29,56)	8:30-10:00	00:15
52(5)	(59,26)	14:00-15:00	1:00
Slutdepot	(30,30)	00:00-99:99	00:00

Tabel 4: Udvidelse af datasættet *Borgerdata* i bilag 1



Figur 7: Løsning på CVRPTWTD1 på Borgerdatasmal

Tidspunkter i tabel 4, starter tidligt og slutter sent, og har overlap mellem morgenhold og aftenhold, for at tillade løsninger af og af hensyn til beregningstid. For at påbyde den tidsmæssige afhængighed mellem borger 1, borger 5 og deres kopier, tilføjes, generalized precedence constraints (34), hvor at

$$\Delta = \{(1, 51), (51, 1)(5, 52), (52, 5)\}$$

og at  $\delta_{1,51} = 60 \cdot 1.5$  og  $\delta_{51,1} = -(3 \cdot 60)$  samt  $\delta_{5,52} = \delta_{52,5} = 0$  (Dohn et al. 2011 [10]). Det var ikke muligt at finde en løsning på problemet på Borgerdata. Grundet resultaterne fra 3.2.5 forsøgte vi på datasættet Borgerdatasmal, med smallere tidsvinduer. Løsningen til problemet ses på Figur 7. Hjælpernummer står i parentes ved siden af borgernummeret. Den optimale løsning på hele problemet ses i Figur 7c, men af hensyn til overskuelighed, plottes de ruter der er fra morgenhold i Figur 7a og aftenhold i Figur 7b. Borger 1, bliver besøgt af samme hjælper, kl 7:30 og igen kl. 09:19, og borger 5 bliver besøgt af hjælper 6 og hjælper 7 kl 15:00. Koden til program samt plot, ses i filen `CVRPTWTD1.py`.

### **3.2.9 Pauser for medarbejdere**

For at overholde pauser kan man yderligere dele morgenhold og aftenhold op i flere tidsintervaler og sørge for at de kommer tilbage til depotet inden en frokostpause eller aftensmad. Der kan medarbejderne spise frokost og diskutere problematikker forbundet med arbejdet, som hjemmesygeplejersken gør i artiklen *Arbejdsdag på 10 timer i hjemmeplejen* (Røge 2017 [28]).

# 4 Resultater

Problem				Resultater				
Model	#K	#H	Sek.	Objektværdi	q	OG	Hold	Data
TSP_MTZ	50	1	5.41	365.25	-	-	-	Borgerdata
TSP_DFJ	50	1	2.50	365.25	-	-	-	Borgerdata
CVRP_MTZ	50	5	1804.19	471.70	11	4.77%	-	Borgerdata
CVRP_Iter	50	5	448.83	471.70	11	-	-	Borgerdata
CVRPTW1	50	5	0.61	605.63	11	-	-	Borgerdata
CVRPTW1	30	3	0.15	407.70	11	-	-	Borgerdata
CVRPTW1	50	5	0.5	825	11	-	-	Borgerdatasmal
CVRPTW1	50	5	107.44	544.16	11	-	-	Borgerdatabred
CVRPTW2	50	5	1800	Ingen løsning	11	-	-	Borgerdata
CVRPTW2	30	3	45.63	407.70	11	-	-	Borgerdata
CVRPTW1_hold	25	3	0.18 <sup>(*)</sup> , 0.19	338.40	11	-	Morgen	Borgerdata
CVRPTW1_hold	25	4	0.86 <sup>(*)</sup> , 0.12	422.17	11	-	Aften	Borgerdata
CVRPTWTD1	52	8	3600	Ingen løsning	8 (fastansat) 6 (vikar)	-	-	Borgerdata
CVRPTWTD1	52	8	41.51	985.27	8 (fastansat) 6 (vikar)	-	-	Borgerdatasmal

#K er antal af kunder og #H er antal af hjælpere i modellen. q angiver kapacitetsbegrænsningen og OG er optimality gap. (\*) angiver kørselstiden for minimering af antal hjælpere.

Tabel 5: Resultater fra modeller kørt i CPLEX 20.1.0.0 på en Macbook Pro 13" M1 2020.

Tabel 5 viser et overblik over opsætning og løsning af præsenterede modeller, samt resultaterne fra 2. Generelt viser resultaterne, at problemer uden tidsvinduer løses hurtigst med de modeller der havde iterativ tilføjelse af begrænsninger, hvorimod problemer med tidsvinduer løses hurtigst med MTZ-lignende begrænsninger. Især havde de præsenterede formuleringer svært ved at finde optimalløsninger til CVRP. Jo flere betingelser der tilføjes til modellerne, desto større bliver den kørte afstand i en optimalløsning. Problemet med tidsmæssige afhængigheder var beregningsmæssigt tung, i forhold til de andre problemer med tidsvinduer. Bredere tidsvinduer i datasættet, giver mulighed for løsninger der har mindre objektværdi, mens smallere vinduer mindsker muligheden og giver en større objektværdi. Smallere tidsvinduer giver kortere beregningstider, og bredere giver længere beregningstider.

# 5 Diskussion

## 5.1 Valg og fravalg

Modellerne er præsenteret i den valgte rækkefølge, dels for at gøre rede for hvordan de matematiske modeller er opstået ud fra planlægningsmodeller der er opstået i virkeligheden, og dels for at følge modelleringsprincippet om at holde modellerne simple. Ved hjemmeplejen er der som nævnt, ofte en dedikeret skemalægger, som kender både plejere og borgerne. Ved at holde modellerne simple kan skemalæggeren ud fra sin erfaring, tage udgangspunkt i resultater fra de enkelte modeller, som har antaget rigide kørselsomkostninger, serviceringstider og varighed af opgaver. Man kunne have valgt at lave en tidsplan over en længere periode, for eksempel 5 dage, en uge eller en måned, hvor der tages højde for at den samme borg bliver besøgt af samme hjælper. Det har vi valgt fra af hensyn til både kompleksiteten af model, samt at der ikke er samlet realistisk data der vil afspejle krav til sådan en planlægning. Derimod kan planlægningen ske på daglig basis med de præsenterede modeller, og hvis besøg, besøgstid og ruter holdes fast, kan man sætte de samme hjælpere på ruterne, for at imødekomme kravet om at borgen bliver besøgt af den samme.

Planlægning på daglig basis vil også være mere robust overfor ændringer i besøg, vagtskifte eller ved tilgang af nye borgere. Yderligere usikkerheder som, for eksempel, trafik bør også være en faktor, men er igen ikke taget højde for grundet manglen på realistisk data. Man kan tilføje en længere servicetid på hver borg for at tilføje en buffer til den næste borg, eller justere omkostningerne på kanterne. Det er her en stor fordel at modellerne er asymmetriske, for eksempel hvis en kant kun benyttes ved myldretid og tidsomkostningen da er asymmetrisk. En mulig og relevant udbygning kunne være at introducere bløde tidsvinduer. Det vil sige, tillade at der må afvigies fra de fastlagte tidsvinduer for at give mere fleksibilitet til planlæggeren såvel som hjælperne. Dette kunne mere konkret indenfor ruteplanlægningen gøres ved at indføre en straf for overskredne tidsvinduer. Her kunne man sætte en maksimal afvigelse der afhænger af opgaven og ellers tildele omkostningen for ruten en straf. Det kan da det gavne ruten at overskride besøgstidernes grænser hvis straffen er lavere end meromkostninger forbundet med at overholde tidsvinduet. Borgene giver da også udtryk i undersøgelsen af PwC (2021 [24]), at forsinkelser er ofte i orden, hvis de får besked inden. Da resultaterne viser at tidsvinduerne spænd stærkt påvirker beregningstiden, vil det have været oplagt at kigge på hvordan disse kan reduceres. Generelt skulle der have været samlet, mere data omkring hjemmeplejen. Vi har valgt at plejerne skal besøge 11 borgere hver i de fleste af optimeringsproblemer baseret på én artikel, men antallet af besøg, vil afhænge af kommune, køretøj, geografi og opgaver.

## 5.2 Implementering og løsning

De iterative metoder var sværest at implementere, da der skal overvejes hvordan brudte begrænsninger identificeres og tilføjes. Implementeringen af (A)TSP og CVRP havde dog god gavn af metoden. Modellerne med MTZ-lignende begrænsninger var lettere at implementere, men kan tage lang tid for CPLEX at bevise til optimalitet. Da modellerne ikke kan tage højde for pludselige hændelser, er optimale løsninger heller ikke strengt nødvendige. Hjemmeplejens skemalæggere kan da have gavn af at tage udgangspunkt i brugbare løsninger til modellerne. De præsenterede iterative implementeringer stopper også kun ved optimale løsninger, og er måske derfor ikke særligt anvendelige for større problemer. Et datasæt der tog højde for at hjemmehjælperne skal holde pause, havde gjort opdeling i morgenhold og aftenhold nemmere. For eksempel ved at der ikke er besøgsvinduerne der overlapper med tidspunktet for frokost og aftensmad. Grundet konstruktionen på *Borgerdata* blev idéerne bag forslagene da lidt vanskeligere at tydeliggøre.

# 6 Konklusion

Når der laves ruteplanlægning i hjemmehjælpen er der mange ting at tage højde for, da man har at gøre med mennesker og menneskelige behov. Behov som for eksempel hjælp til madlavning, rengøring eller personlig pleje, men også behov som for eksempel, medicinering ellerrensning af sår. Derudover er det også vigtigt at tage højde for borgernes psykiske behov ved for eksempel at tildele borgeren færre faste hjælpere i stedet for mange forskellige, og sørge at borgeren ved hvornår der foretages besøg. I hjemmehjælpen bliver ruterne planlagt af en ruteplanlægger som forsøger at tage højde for alle borgernes behov.

Med udgangspunkt i kurset *Modellering og løsning af optimeringsproblemer* er der, ud fra de fundne behov og retningslinjer indenfor hjemmehjælpen, undersøgt og opstillet matematiske modeller, som kan være med til at hjælpe og optimere ruteplanlægningen i hjemmehjælpen. De matematiske modeller er allerede kendte modeller, som er opstillet, forklaret og derefter implementeret med udgangspunkt i virkelighedsproblemet. Der er bl.a. kigget nærmere på Traveling Salesman problem (TSP), Capacitated Vehicle Routing Problem (CVRP) og Capacitated Vehicle Routing Problem with Time Windows (CVRPTW) og opstillet modeller for disse. Derudover er der undersøgt forskellige metoder til eliminering af subtur og tilføjet yderligere begrænsninger til, for eksempel, at tage højde for synkronisering eller begrænse antallet af borgere per hjælper. De fleste modeller er implementeret i Python og løst på eget data. Datasættet er manuelt konstrueret men med udgangspunkt i virkelige scenarier, og det forsøger derfor at repræsentere virkelig data så vidt muligt.

Resultaterne har vist at (A)TSP, samt CVRP formuleringer på vores datasæt hurtigst kan løses med DFJ-SEC, hvor subtur elimineringsbegrænsninger iterativt tilføjes til modellen. Modellerne med tidsvinduer løses her bedst med MTZ-lignende begrænsninger, i stedet for vores naive iterative metode og er nemmere at implementere. Resultaterne viser at kortere tidsvinduer er til gavn for løsningstid samtidig med at det er hensigtsmæssigt for borgeren. Der er gennemgået praktiske overvejelser, som for eksempel sygdom, vikarer, arbejdstider, afhængigheder mellem besøg og pauser for hjemmehjælpere, samt måder at imødekomme dem.

# Litteratur

- [1] *The traveling salesman problem : a guided tour of combinatorial optimization.* Wiley-Interscience series in discrete mathematics. Wiley, Chichester, 1985, ISBN 0471904139.
- [2] Ascheuer, Norbert, Fischetti, Matteo og Grötschel, Martin: *Solving the Asymmetric Traveling Salesman Problem with time windows by branch-and-cut.* Mathematical Programming, 90(3):475–506, 2001, ISSN 1436-4646. <https://doi.org/10.1007/PL00011432>.
- [3] Bard, Jonathan F., Kontoravdis, George og Yu, Gang: *A Branch-and-Cut Procedure for the Vehicle Routing Problem with Time Windows.* Transportation Science, 36(2):250–269, 2002, ISSN 00411655, 15265447. <http://www.jstor.org/stable/25769106>, besøgt den 2022-06-11.
- [4] Bektas, Tolga: *The multiple traveling salesman problem: an overview of formulations and solution procedures.* Omega, 34(3):209–219, 2006, ISSN 0305-0483. <https://www.sciencedirect.com/science/article/pii/S0305048304001550>.
- [5] Bredström, David og Rönnqvist, Mikael: *Combined vehicle routing and scheduling with temporal precedence and synchronization constraints.* European Journal of Operational Research, 191(1):19–31, 2008, ISSN 0377-2217. <https://www.sciencedirect.com/science/article/pii/S0377221707007436>.
- [6] Danmarks statistik: *Flere personer visiteres til hjemmehjælp.* Report 206, Danmarks Statistik, 10. juni 2022. <https://www.dst.dk/da/Statistik/nyheder-analyser-publ/nyt/NytHtml?cid=38425>, besøgt den 10-06-2022.
- [7] Dantzig, G., Fulkerson, R. og Johnson, S.: *Solution of a Large-Scale Traveling-Salesman Problem.* Journal of the Operations Research Society of America, 2(4):393–410, 1954, ISSN 00963984. <http://www.jstor.org.ez.statsbiblioteket.dk:2048/stable/166695>, besøgt den 30-05-2022.
- [8] Dantzig, G. B og Ramser, J. H: *The Truck Dispatching Problem.* Management science, 6(1):80–91, 1959, ISSN 0025-1909.
- [9] Desrochers, Martin og Laporte, Gilbert: *Improvements and extensions to the Miller-Tucker-Zemlin subtour elimination constraints.* Operations research letters, 10(1):27–36, 1991, ISSN 0167-6377.
- [10] Dohn, Anders, Rasmussen, Matias Sevel og Larsen, Jesper: *The vehicle routing problem with time windows and temporal dependencies.* Networks, 58(4):273–289, 2011. <https://onlinelibrary.wiley.com/doi/abs/10.1002/net.20472>.

- [11] Drexel, Michael: *Synchronization in Vehicle Routing—A Survey of VRPs with Multiple Synchronization Constraints.* Transportation Science, 46(3):297–316, 2012, ISSN 00411655, 15265447. <http://www.jstor.org/stable/23263544>, besøgt den 2022-06-11.
- [12] IBM: *CPLEX® Optimizers*, 03-05-2021 2019. <https://www.ibm.com/docs/en/icos/12.9.0?topic=cplex>.
- [13] Johnson, Donald B.: *Finding All the Elementary Circuits of a Directed Graph*. SIAM Journal on Computing, 4(1):77–84, 1975. <https://doi.org/10.1137/0204007>.
- [14] Kallehauge, Brian: *Formulations and exact algorithms for the vehicle routing problem with time windows*. Computers Operations Research, 35(7):2307–2330, 2008, ISSN 0305-0548. <https://www.sciencedirect.com/science/article/pii/S0305054806002929>, Part Special Issue: Includes selected papers presented at the ECCO'04 European Conference on combinatorial Optimization.
- [15] Kallehauge, Brian, Larsen, Jesper, Madsen, Oli B.G. og Solomon, Marius M.: *Vehicle Routing Problem with Time Windows*, sider 67–98. Springer US, Boston, MA, 2005, ISBN 978-0-387-25486-9. [https://doi.org/10.1007/0-387-25486-2\\_3](https://doi.org/10.1007/0-387-25486-2_3).
- [16] Laporte, Gilbert: *The traveling salesman problem: An overview of exact and approximate algorithms*. European Journal of Operational Research, 59(2):231–247, 1992, ISSN 0377-2217. <https://www.sciencedirect.com/science/article/pii/037722179290138Y>, besøgt den 31-05-2022.
- [17] Letchford, Adam N. og Salazar-González, Juan José: *Projection results for vehicle routing*. Mathematical Programming, 105(2-3):251, Februar 2006. <https://www.proquest.com/scholarly-journals/projection-results-vehicle-routing/docview/232850107/se-2?accountid=14468>, Copyright - Springer-Verlag Berlin Heidelberg 2006; Last updated - 2021-09-11; CODEN - MHPGA4; SubjectsTermNotLitGenreText - United States-US.
- [18] Little, John D. C., Murty, Katta G., Sweeney, Dura W. og Karel, Caroline: *An Algorithm for the Traveling Salesman Problem*. Operations Research, 11(6):972–989, 1963. <https://pubsonline.informs.org/doi/abs/10.1287/opre.11.6.972>, besøgt den 30-05-2022.
- [19] Miller, C., Tucker, A. og Zemlin, R.: *Integer Programming Formulation of Traveling Salesman Problems*. Journal of the ACM, 7(4):326–329, 1960, ISSN 0004-5411.
- [20] NetworkX: *simple\_cycles*, 2021. [https://networkx.org/documentation/stable/reference/algorithms/generated/networkx.algorithms.cycles.simple\\_cycles.html](https://networkx.org/documentation/stable/reference/algorithms/generated/networkx.algorithms.cycles.simple_cycles.html), besøgt den 14-06-2022.

- [21] Pferschy, Ulrich og Staněk, Rostislav: *Generating subtour elimination constraints for the TSP from pure integer solutions*. Central European journal of operations research, 25(1):231–260, 2016, ISSN 1435-246X.
- [22] Pidd, Michael: *Just Modeling Through: A Rough Guide to Modeling*. INFORMS Journal on Applied Analytics, 29(2):118–132, 1999. <https://pubsonline.informs.org/doi/abs/10.1287/inte.29.2.118>, besøgt den 30-05-2022.
- [23] PuLP: *PuLP 2.6.0, Project description*, 2021. <https://pypi.org/project/PuLP/>, besøgt den 14-06-2022.
- [24] PwC for Videnscenter for Værdig Ældrepleje: *Sårbare ældres møde med hjemmeplejen*. Rapport, Sundhedsstyrelsen, 2021. [https://www.sst.dk/-/media/Udgivelser/2021/AEldre/Saarbare-aeldres-moede-med-hjemmeplejen/Saarbare-aeldres-moede-med-hjemmeplejen\\_Rapport\\_2021.ashx?sc\\_lang=da&hash=37C76074266CA71EE93EFD89F1408433](https://www.sst.dk/-/media/Udgivelser/2021/AEldre/Saarbare-aeldres-moede-med-hjemmeplejen/Saarbare-aeldres-moede-med-hjemmeplejen_Rapport_2021.ashx?sc_lang=da&hash=37C76074266CA71EE93EFD89F1408433), besøgt den 13-06-2022.
- [25] Raff, Samuel: *Routing and scheduling of vehicles and crews: The state of the art*. Computers Operations Research, 10(2):63–211, 1983, ISSN 0305-0548. <https://www.sciencedirect.com/science/article/pii/0305054883900308>, Routing and Scheduling of Vehicles and Crews. The State of the Art.
- [26] Rambøll: *Undersøgelse køretider i hjemmeplejen*. Rapport, November 2018.
- [27] Rasmussen, Matias Sevel, Justesen, Tor, Dohn, Anders og Larsen, Jesper: *The Home Care Crew Scheduling Problem: Preference-based visit clustering and temporal dependencies*. European Journal of Operational Research, 219(3):598–610, 2012, ISSN 0377-2217. <https://www.sciencedirect.com/science/article/pii/S0377221711009891>, Feature Clusters.
- [28] Røge, Pia Lüders: *Arbejdssdag på 10 timer i hjemmeplejen*. Sygeplejersken, (Nr. 2), 2017. <https://dsr.dk/sygeplejersken/arkiv/sy-nr-2017-2/arbejdssdag-paa-10-timer-i-hjemmeplejen>, besøgt den 30-05-2022.
- [29] Styrelsen for Patientsikkerhed, Tilsyn og Rådgivning Nord: *Tilsynsrapport, Hjemmepleje Odder Kommune*. Tilsynsrapport, Styrelsen for Patientsikkerhed, 2022. <https://stps.dk/da/tilsyn/tilsynsrapporter/~/media/0A4558A9621A418681910362EEC65E63>, besøgt den 13-06-2022.
- [30] Toth, Paolo og Vigo, Daniele: *The Vehicle Routing Problem*. Society for Industrial and Applied Mathematics, USA, 2001, ISBN 0898714982.
- [31] Williams, H. Paul: *Model Building in Mathematical Programming*. Wiley, New York, 5. udgave, 2013, ISBN 9781118443330.

# Bilag

## Bilag 1: Datasæt

Tabel 6: Datasættet *Borgerdata*.

Navn	Adresse	Besøgsvindue	Servicetid
Depot	(30,30)	00:00-99:99	00:00
1	(29,56)	07:30-08:00	00:30
2	(10,1)	11:00-13:00	00:20
3	(2,2)	16:00-17:00	00:20
4	(25,18)	08:00-10:00	00:10
5	(59,26)	14:00-15:00	01:00
6	(56,52)	16:00-16:30	00:30
7	(30,10)	07:00-08:00	00:15
8	(15,48)	08:00-13:00	00:40
9	(13,47)	12:00-16:00	00:30
10	(10,31)	16:00-21:00	00:20
11	(1,14)	17:30-19:00	01:00
12	(43,49)	18:00-20:00	00:20
13	(23,54)	09:00-13:00	00:10
14	(22,15)	08:00-12:00	00:25
15	(27,43)	12:00-16:00	00:30
16	(21,48)	15:00-17:00	00:20
17	(51,18)	11:00-14:00	00:30
18	(45,12)	07:30-08:00	00:20
19	(51,35)	11:00-13:00	00:10
20	(53,26)	18:00-20:00	00:15
21	(21,34)	11:00-14:00	00:30
22	(52,40)	16:00-21:00	00:20
23	(35,15)	07:00-08:00	00:20
24	(20,51)	08:00-10:00	00:30
25	(0,31)	18:00-20:00	00:10
26	(10,55)	11:00-13:00	01:00
27	(48,25)	16:00-21:00	00:20
28	(14,57)	08:00-10:00	00:30
29	(18,15)	07:30-08:00	00:20
Fortsætter på næste side			

**Tabel 6 – fortsat fra forrig side**

Navn	Adresse	Besøgsvindue	Servicetid
30	(39,24)	11:00-13:00	00:15
31	(15,1)	08:00-10:00	00:20
32	(59,45)	11:00-13:00	00:30
33	(42,17)	16:00-21:00	00:15
34	(44,30)	18:00-20:00	00:20
35	(41,11)	07:30-08:00	00:10
36	(10,51)	08:00-12:00	00:10
37	(31,55)	08:00-12:00	00:15
38	(46,37)	09:00-13:00	00:20
39	(36,6)	08:00-10:00	00:15
40	(8,18)	18:00-20:00	00:20
41	(58,1)	14:00-15:00	01:00
42	(37,19)	12:00-16:00	00:20
43	(41,43)	14:00-15:00	00:10
44	(35,53)	12:00-16:00	01:00
45	(4,15)	09:00-13:00	00:10
46	(51,7)	16:00-21:00	00:15
47	(39,25)	14:00-15:00	00:20
48	(49,47)	18:00-20:00	00:10
49	(28,3)	12:00-16:00	00:15
50	(23,46)	16:00-21:00	00:15

Bilag 2: Jens Lysgaard 2021, *Modellering og løsning af optimeringsproblemer, slides*

## Kapitel 8: Heltalsprogrammering

Anvendelighed af  
heltalsprogrammer-  
ing  
Løsningsmetoder  
til heltalsmodelle



H. P. Williams.  
*Model Building in Mathematical Programming*, 5. udg.  
Wiley & Sons, 2013.

Jens Lysgaard  
Modellering og løsning af optimeringsproblemer, 2021

## Diskrete input og output

Figurerne er en grafisk afbilledning af følgende LP-problem:  
(fra s. 156 i bogen)

$$\begin{array}{l} \text{max.} \\ \text{s.t.} \\ -2x_1 + 2x_2 \\ -8x_1 + 10x_2 \\ | \wedge | \vee \\ 13 \end{array}$$

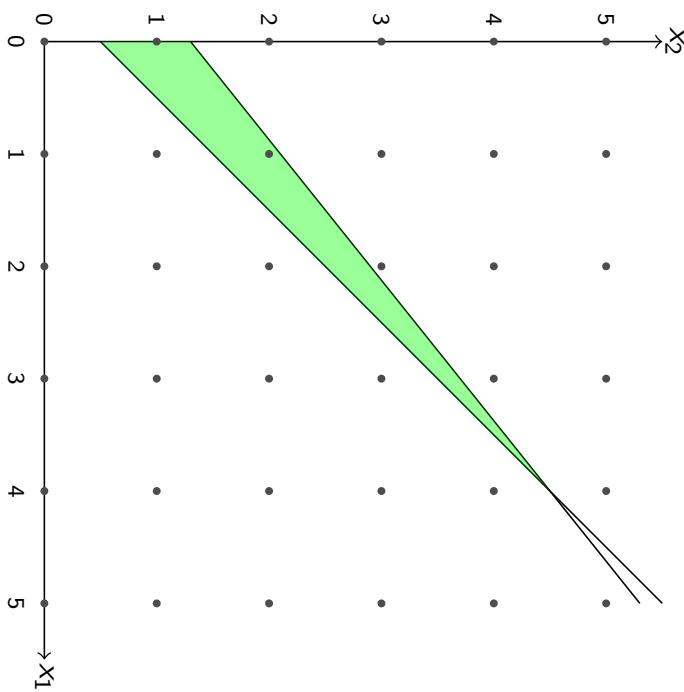
I figuren er også markeret alle (brugbare og ikke-brugbare)

Optimaløsningen til LP'et er i punktet (4, 4,5)

Den optimale heltalsløsning er i punktet (1, 2)

I nogle tilfælde kan det være næriggende blot at foretage arrundning af LP-løsningen til heltal, hvis man reelst ønsker en heltalsløsning ikke findes blot ved afrunding.

I sadanlænne tilfælde er der tale om behovet for at formulere og løse en model med heltallige variable



Anvendelighed af  
heltalsprogrammer-  
ing

Diskrete input og output

Logiske betingelser

Kombinatoriske problemer

Ikke-lineære problemer

## Logiske betingelser

- ▼ Logiske betingelser involverer generelt en skelnen mellem, om et udtryk er sandt eller falsk, og konsekvenserne heraf
- ▼ Beslutningsvariable, som kun kan antage værdierne 0 og 1, kan benyttes til at repræsentere hhv. falsk og sandt
- ▼ Generelt giver dette mulighed for at modellere 'Hvis-så' sammenhænge
- ▼ Som ét blandt rigtig mange eksempler kan en 0-1 variabel benyttes til at angive, om man beslutter sig for at placere en fabrik i en given lokation
- ▼ Som konsekvens af denne beslutning kan der opstå mulighed for at foretage levering fra den pågældende lokation, for så vidt at fabrikken placeres der
- ▼ I så fald vil man have behov for at begrænsning, som udtrykker, at den leverede mængde fra fabrikken skal være nul, hvis fabrikken ikke oprettes, hvorimod den leverede mængde kan være positiv, hvis fabrikken oprettes
- ▼ Den tilhørende logik i form af en begrænsning kan håndteres vha. bl.a. en 0-1 variabel, som angiver, om fabrikken oprettes

Anvendelighed af  
heltalsprogrammer-  
ing  
Diskrete input og output  
**Logiske betingelser**  
Kombinatoriske problemer  
Ikke-lineære problemer  
Netværksproblemer  
**Løsningsmetoder**  
til heltalsmodeller

# Kombinatoriske problemer

- ▼ Kombinatoriske optimeringsproblemer er en fællesbetegnelse for optimeringsproblemer, som drejer sig om at bestemme en optimal løsning i et endeligt løsningsrum, dvs. at der er et endeligt antal løsninger at vælge mellem
- ▼ Det endelige antal løsninger er dog typisk meget stort, hvilket normalt gør det til en stor udfordring at bestemme en løsning, som med sikkerhed er optimal
- ▼ Et kombinatorisk optimeringsproblem kan i mange tilfælde involvere én af følgende dele:
  - ▼ Udvælgelse af en delmængde
  - ▼ Gruppering af elementer
  - ▼ Fordeling af opgaver mellem resourcer
  - ▼ Bestemmelse af rækkefølger, evt. i kombination med én af ovenstående muligheder

Anvendelighed af  
heltalsprogrammer-  
ing

Diskrete input og output

Logiske betingelser

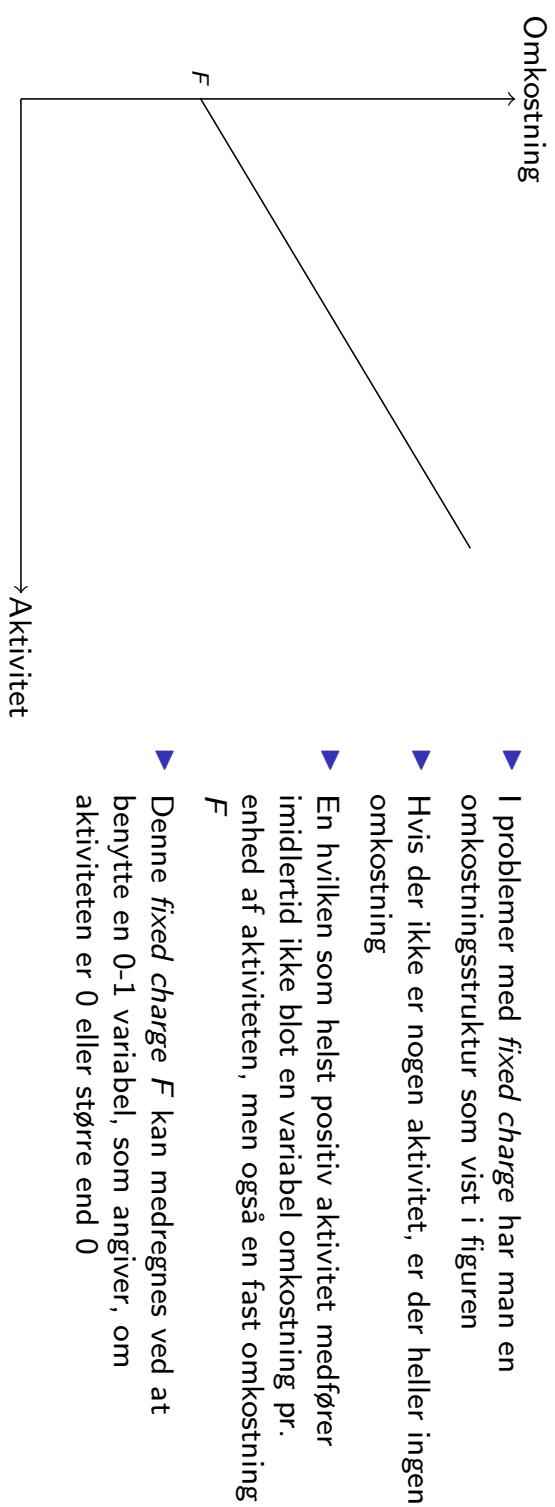
Kombinatoriske problemer

Ikke-lineære problemer

Netværksproblemer

Løsningsmetoder  
til heltalsmodeller

## Ikke-lineære problemer: Fixed charge



Anvendelighed af  
heltalsprogrammer-  
ing

Diskrete input og output

Logiske betingelser

Kombinatoriske problemer

Ikke-lineære problemer

Netværksproblemer

Løsningsmetoder  
til heltalsmodeller

# Ikke-lineære problemer: Det kvadratiske assignment problem

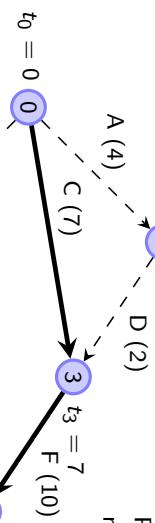
- 
- I det kvadratiske assignment problem (QAP: Quadratic Assignment Problem) er der givet  $n$  fabrikker  $F_1, \dots, F_n$  og  $n$  lokationer  $L_1, \dots, L_n$
- For hvert par af fabrikker  $\{i, j\}$  er givet en mængde  $f_{ij}$ , som skal sendes mellem de to fabrikker
  - For hvert par af lokationer  $\{i, j\}$  er givet en afstand  $d_{ij}$  mellem de to lokationer
  - Problemet går ud på at tildele fabrikker til lokationer, således at de samlede omkostninger minimeres, summeret over alle par af lokationer
  - Omkostningen mellem hvert par af lokationer er givet ved flow gange afstand, som vist i figuren mellem  $L_1$  og  $L_2$
  - Hvis der benyttes samme variable som i det lineære Assignment Problem, er omkostningsfunktionen ikke lineær men derimod kvadratisk, idet omkostningen mellem lokation  $i$  og  $j$  afhænger af både hvilken fabrik der placeres i lokation  $L_i$ , og hvilken fabrik der placeres i lokation  $L_j$
  - I eksemplet er omkostningen mellem  $L_1$  og  $L_2$  lig med  $f_{34}d_{12}$ , hvilket er under forudsætning af både  $x_{31} = 1$  og  $x_{42} = 1$ , hvor  $x_{ij} = 1$  hvis  $F_i$  er tildelt lokation  $L_j$ , ellers er  $x_{ij} = 0$

Anvendelighed af  
heltalsprogrammer-  
ing  
Diskrete input og output  
Logiske betingelser  
Kombinatoriske problemer  
**Ikke-lineære problemer**  
Netværksproblemer  
Løsningsmetoder  
til heltalsmodeller

# Netværksproblemer: Ressourceallokering i projektplanlægning

$$t_1 = 5$$

Projekthetværket i bogens figur 5.8  
med aktiviteter på kanter og varigheder i parentes



Anvendelighed af  
heltalsprogrammer-  
ing  
Diskrete input og output  
Logiske betingelser  
Kombinatoriske problemer  
Ikke-lineære problemer  
Netværksproblemer  
Løsningsmetoder  
til heltalsmodeller

- ▼ Ved brug af LP (uden krav om heltalsvariable) kan identificeres den kritiske vej (0,3,4,2,5,6) i netværket
- ▼ Derudover kan der være behov for yderligere analyser, som alle involverer heltalsprogrammering, f.eks.:
  - ▼ En analyse af, hvorledes ressourcer bedst fordeles mellem aktiviteter over tid, hvis flere aktiviteter bruger af de samme ressourcer, f.eks. mannskab
  - ▼ En analyse af, hvilke aktiviteter det vil være mest nyttigt at kunne reducere varigheden af, evt. i kombinationer af aktiviteter

## Netværksproblemer: Andre heltalsmodeller

- ▼ Udover projektplanlægning kan der mere generelt opstå behov for heltalsmodeller som udvidelse af LP-modeller, f.eks. i forbindelse med flg.:
  - ▶ I transportproblemets: Etablering af nye udbydere i netværket (f.eks. opførelse af fabrikker)
  - ▶ I transhipmentproblemets: Udvidelse af kapaciteten af transhipmentpunkter, f.eks. i distributionscentre
  - ▶ I maximum flow problemet: Udvidelse af kapaciteten på kanter, f.eks. i trafikplanlægning

Anvendelighed af  
heltalsprogrammer-  
ing  
Diskrete input og output  
Logiske betingelser  
Kombinatoriske problemer  
Ikke-lineære problemer  
Netværksproblemer  
Løsningsmetoder  
til heltalsmodeller

Anvendelighed af  
heltalsprogrammer-  
ing

Løsningsmetoder  
til heltalsmodeller

Sniplansmetoder  
Branch and bound metoder

## Løsningsmetoder til heltalsprogrammering

# Snitplansmetoder: Pseudo-kode

---

**En generisk snitplansmetode**

---

```
// Input: En ILP model
1 Konstruer LP relaksationen af ILP'et ved at erstatte heltalskrav med bounds
2 repeat
3   Løs LP relaksationen og lad  $x^*$  betegne den optimale løsning til LP'et
4   if  $x^*$  er heltallig then
5      $x^*$  er optimalløsning til ILP'et
6   else
7     Identificér en lineær ulighed, som overholdes af alle brugbare heltalsløsninger, men som
     ikke overholdes af  $x^*$ . Tiføj denne lineære ulighed til LP'et
8   end
9 until  $x^*$  er heltallig
```

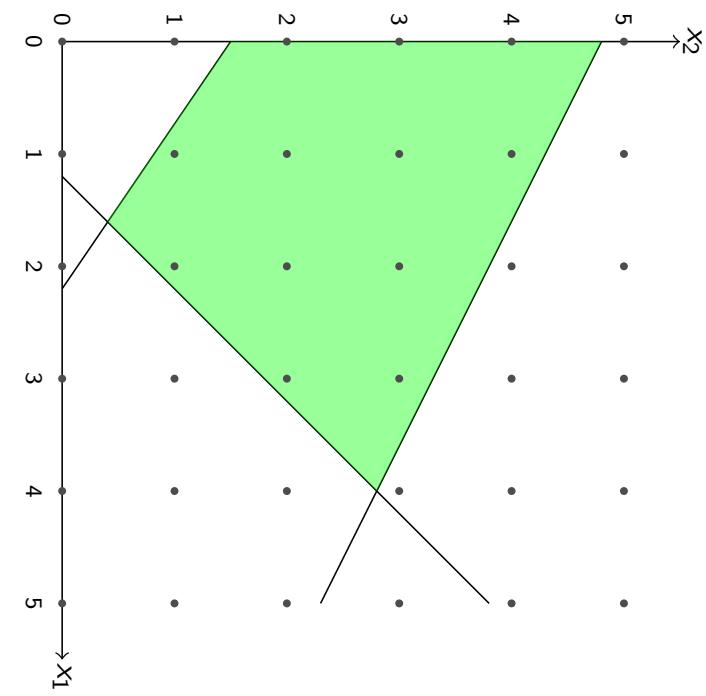
---

Anvendelighed af  
heltalsprogrammer-  
ing

Løsningsmetoder  
til heltalsmodeller

Snitplansmetoder  
Branch and bound metoder

## Snitplansmetoder: Grafisk illustration



- ▼ Figuren er en gengivelse af Figur 10.1 på s. 211 i bogen
- ▼ Det grønne område repræsenterer det brugbare område i et LP
- ▼ Hvis det også er krævet, at både  $x_1$  og  $x_2$  skal være heltallige, så er mængden af brugbare løsninger begrænset til de markede punkter i det grønne område
- ▼ Hvis man i det viste eksempel løser et LP over det grønne område, så får man en ikke-heltallig løsning, idet alle hjørnepunkter i det grønne område er ikke-heltallige
- ▼ Snitplansmetoder tager udgangspunkt i hele det grønne område og tilføjer derefter iterativt lineære begrænsninger, som hver især eliminerer en del af det grønne område, indeholdende den aktuelle LP-løsning, fra det brugbare område, men uden at eliminere heltalspunkter

Anvendelighed af  
heltalsprogrammer-  
ing

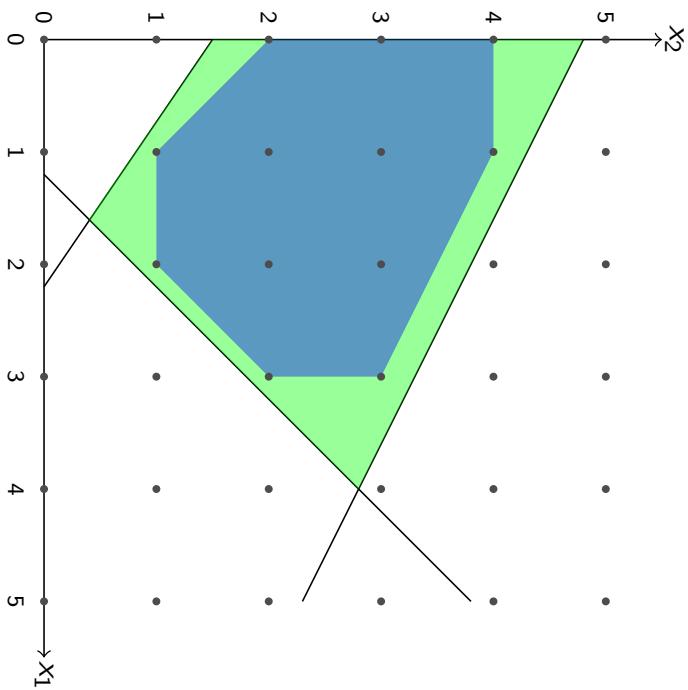
Løsningsmetoder  
til heltalsmodeller

Snitplansmetoder

Branch and bound metoder

# Snitplansmetoder: Det konvekse hylster

- ▼ Det grønne område repræsenterer det brugbare område i et LP
- ▼ Det blå område viser det konvekse hylster af alle brugbare heltalspunkter (den mindste konvekse mængde, som indeholder alle brugbare heltalspunkter)
- ▼ Hvis man kender de lineære begrænsninger, der definerer det blå område, så kan man optimere LP'et inkl. disse begrænsninger og med sikkerhed få en heltallig LP-løsning
- ▼ Generelt kan det være en meget omfattende opgave at identificere disse lineære begrænsninger
- ▼ Snitplansmetoder kan dog også ende med et større område end det blå område, når blot den optimale LP-løsning er heltallig
- ▼ Ydermere kan snitplansmetoder blot bruges til at reducere det grønne område i hvert delproblem i en branch and bound metode, inden der foretages branching fra delproblemet. Dette er generelt ideen i *branch and cut* metoder



Anvendelighed af  
heltalsprogrammer-  
ing  
Løsningsmetoder  
til heltalsmodeller  
Snitplansmetoder  
Branch and bound metoder

## Branch and bound: Branching

- ▶ Vi kan generelt betragte et optimeringsproblem, der er formuleret således:

$$\begin{array}{ll}\text{min.} & c'x \\ \text{s.t.} & x \in F\end{array}$$

- ▶ Mængden  $F$  er således mængden af brugbare løsninger til problemet
- ▶ Vi kan generelt opdele en mængde  $F$  af brugbare løsninger i  $k$  delmængder  $F_1, \dots, F_k$ , således at  $F_1 \cup \dots \cup F_k = F$
- ▶ Dette har som konsekvens, at den optimale løsning til  $F$  vil findes i én eller flere af delmængderne
- ▶ De enkelte par af delmængder kan evt. have en ikke-tom fællesmængde, men typisk har vi  $F_i \cap F_j = \emptyset$  for alle  $i, j = 1, \dots, k$ ,  $i \neq j$

Anvendelighed af  
heltalsprogrammer-  
ing  
Løsningsmetoder  
til heltalsmodeller  
Sniplansmetoder  
Branch and bound metoder

## Branch and bound: Bounding

- ▼ For en vilkårlig delmængde  $F_i$  af brugbare løsninger kan vi definere en *lower bound*  $b(F_i)$  således:
- $$b(F_i) \leq \min_{x \in F_i} c'x$$
- ▼ Valget af metode til beregning af lower bounds involverer en balance mellem løsningskvalitet (dvs. en værdi tæt på den optimale objektfunktionsværdi) og beregningsomfang
- ▼ Nyttet af en lower bound kommer til udtryk ved, at hvis vi kender en brugbar løsning  $x$  med objektfunktionsværdi  $U$ , så vil  $b(F_i) \geq U$  være tilstrækkeligt til at sikre, at vi ikke kan finde en bedre løsning i delmængden  $F_i$ , som derfor ikke behøver at blive undersøgt yderligere

Anvendelighed af  
heltalsprogrammer-  
ing

Løsningsmetoder  
til heltalsmodeller

Snitplansmetoder

Branch and bound metoder

## Branch and bound: Kombinationer af branching og bounding

- ▼ Generelt kan der dannes mange varianter af branch and bound algoritmer ved forskellige kombinationer af elementer til hhv. branching og bounding
- ▼ I mange tilfælde frembringes en lower bound som resultatet (objektfunktionsværdien) af et optimeringsproblem. I så fald vil man typisk foretage branchingen, således at optimaløsningen på et problem  $F_i$  er ikke-brugbart til hvert af  $F_i$ 's delproblemer
  - ▼ Hvis man anvender Assignment Problemet til at frembringe lower bounds for et Asymmetrisk Traveling Salesman Problem, så kan man branche ved at forbyde kanter i en subtour, således at assignmentløsningen til  $F_i$  bliver ikke-brugbar i hvert af  $F_i$ 's delproblemer
  - ▼ Hvis man anvender LP relaksationen af et ILP til at frembringe lower bounds, så kan man branche på en enkelt fraktionel variabel, således at LP-løsningen til  $F_i$  bliver ikke-brugbar i hvert af  $F_i$ 's delproblemer

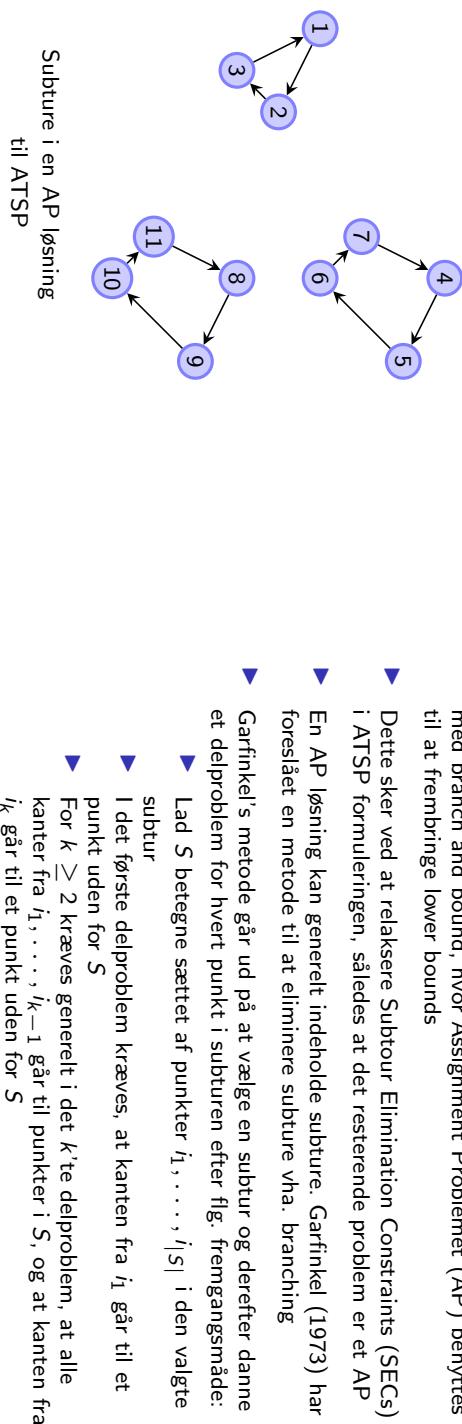
Anvendelighed af  
heltalsprogrammer-  
ing  
Løsningsmetoder  
til heltalsmodeller  
Snitplansmetoder  
Branch and bound metoder

## Branch and bound: Søgestrategier

- ▼ Branch and bound involverer generelt, at man vedligeholder et søgetræ, som indeholder de genererede delproblemer
- ▼ Søgetræet kan undersøges med brug af forskellige strategier, f.eks.:
  - ▼ Depth-First Search (DFS) involverer, at man så vidt muligt søger nedad i træet. Dette begrænser antallet af samtidige aktive delproblemer (dvs. delproblemer, som endnu ikke er færdigbehandlede)
  - ▼ Breadth-First Search (BFS) involverer, at man først genererer alle delproblemer på ét niveau, inden man går videre til næste niveau
  - ▼ Best Bound Search (BBS) involverer, at man i hver iteration brancher fra det delproblem, som har den bedste bound (dvs. den mindste lower bound i et minimeringsproblem)
- ▼ DFS er typisk *relativt nem* at implementere og kræver kun begrænset computerhukommelse, men kan involvere lange beregningstider. Derimod kan BBS være mere kompliceret at implementere og kan kræve mere computerhukommelse pga. mange aktive subproblemer men kan samtidig være beregningsmæssigt mere effektiv

Anvendelighed af  
heitalsprogrammer-  
ing  
Løsningsmetoder  
til heitalsmodeller  
Snitplansmetoder  
Branch and bound metoder

# Branching med AP bounds til ATSP



R. S. Garfinkel.

On partitioning the feasible set in a branch-and-bound algorithm for the asymmetric traveling-salesman problem.  
*Operations Research*, 21(1):340–343, 1973.

Anvendelighed af  
heltalsprogrammer-  
ing  
Løsningsmetoder  
til heltalsmodeller  
Sniplansmetoder  
Branch and bound metoder

- ▼ Det Asymmetriske Traveling Salesman Problem (ATSP) kan løses med branch and bound, hvor Assignment Problem (AP) benyttes til at frembringe lower bounds
- ▼ Dette sker ved at relaksere Subtour Elimination Constraints (SECs) i ATSP formuleringen, således at det resterende problem er et AP
- ▼ En AP løsning kan generelt indeholde subture. Garfinkel (1973) har foreslået en metode til at eliminere subture vha. branching
- ▼ Garfinkel's metode går ud på at vælge en subtur og derefter danne et delproblem for hvert punkt i subturen efter flg. fremgangsmåde:
  - ▼ Lad  $S$  betegne sættet af punkter  $i_1, \dots, i_{|S|}$  i den valgte subtur
  - ▼ I det første delproblem kræves, at kanten fra  $i_1$  går til et punkt uden for  $S$
  - ▼ For  $k \geq 2$  kræves generelt i det  $k$ 'te delproblem, at alle kanter fra  $i_1, \dots, i_{k-1}$  går til punkter i  $S$ , og at kanten fra  $i_k$  går til et punkt uden for  $S$

# Branching på en subtur i AP relaksationen til ATSP

S

$\bar{S}$

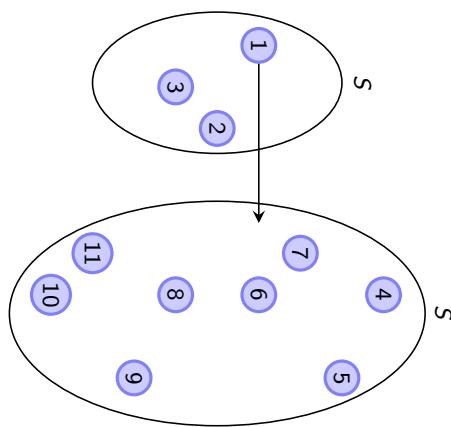
S

$\bar{S}$

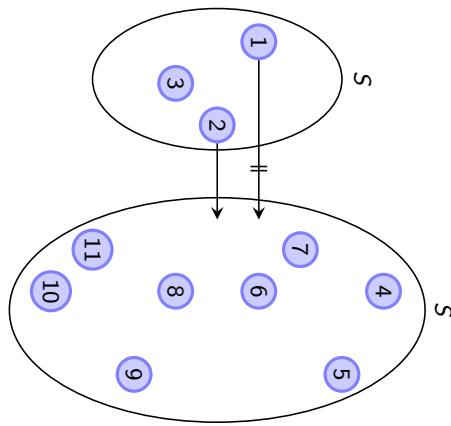
S

$\bar{S}$

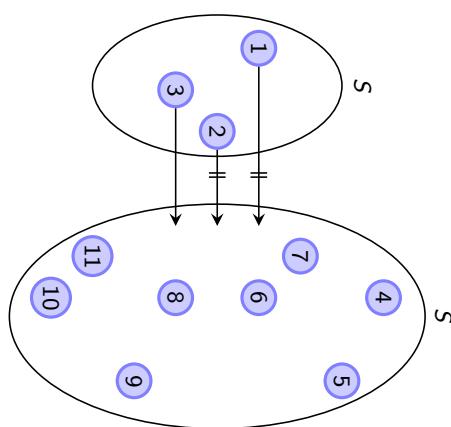
Delproblem 1



Delproblem 2



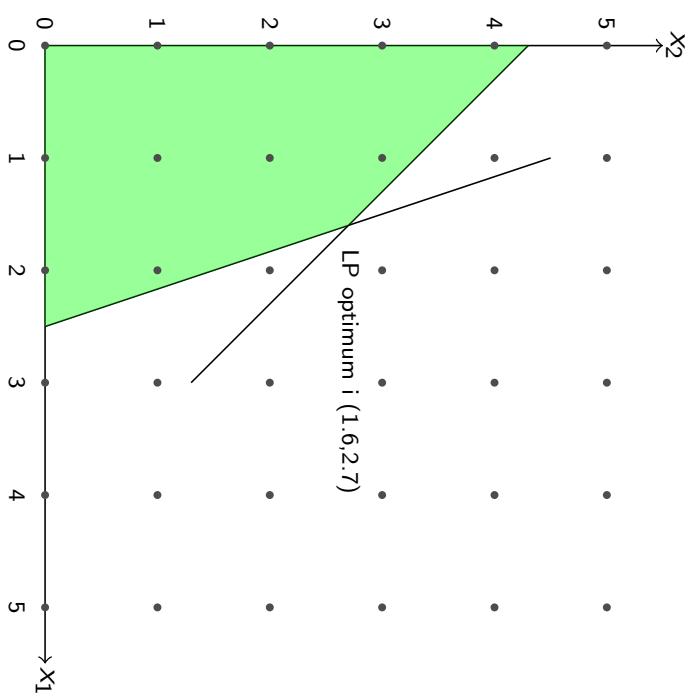
Delproblem 3



Anvendelighed af  
heitalsprogrammer-  
ing  
Løsningsmetoder  
til heitalsmodeller  
Sniplansmetoder  
Branch and bound metoder

- ▼ I eksemplet branches på en subtur for punktmængden  $S = \{1, 2, 3\}$
- ▼ Punktmængden uden for  $S$  betegnes  $\bar{S}$
- ▼ I det  $k$ 'te delproblem kræves en kant fra punkt  $i_k \in S$  til  $\bar{S}$ , samtidig med at alle kanter fra  $i_1, \dots, i_{k-1} \in S$  til  $\bar{S}$  udelukkes
- ▼ Dette sikrer, at enhver brugbar ATSP løsning findes i netop ét af delproblemerne, samtidig med at den aktuelle AP løsning med subturen i  $S$  elimineres i hvert delproblem
- ▼ Udelukkelsen af kanter sker ved at sætte deres omkostning til uendelig, således at også delproblemerne er AP'er. Kravet om en kant fra  $i_k$  ud af  $S$  sker ved at udelukke kanter fra  $i_k$  til andre punkter i  $S$

# Branching med LP bounds til ILP



- ▼ Et Integer Linear Programming (ILP) problem kan løses med branch and bound, hvor LP relaksationen benyttes til at frembringe lower bound
- ▼ LP relaksationen fremkommer ved at erstatte heltalskrav med bounds på variable
- ▼ Løsningen på LP'et udgør en lower bound (i et minimeringsproblem) for værdien af objektfunktionen i den optimale heltalsløsning
- ▼ Hvis LP løsningen er fraktionel, kan branching foretages på en enkelt fraktionel variabel ved at danne to delproblemer
- ▼ Lad  $x$  være den variabel, der branches på, og  $x^*$  dens aktuelle fraktionelle værdi
- ▼ I det ene delproblem tilføjes begrænsningen  $x \leq \lfloor x^* \rfloor$
- ▼ I det andet delproblem tilføjes begrænsningen  $x \geq \lceil x^* \rceil$
- ▼ Hvis der i eksemplet branches på  $x_1$ , vil man tilføje begrænsningerne henholdsvis  $x_1 \leq 1$  og  $x_1 \geq 2$
- ▼ Hvis der i eksemplet branches på  $x_2$ , vil man tilføje begrænsningerne henholdsvis  $x_2 \leq 2$  og  $x_2 \geq 3$

Anvendelighed af  
heltalsprogrammer-  
ing  
Løsningsmetoder  
til heltalsmodeller  
Sniplansmetoder  
Branch and bound metoder

## Bilag 3: Kodeoversigt

Dette er en oversigt over filerne i `Kodebilag.zip`, som er afleveret separat.

Til hver fil, er en tilhørende 'CPLEX.log' samt to CPLEX-filer; et problem i '.lp'-format og en løsning i '.sol'-format.

Et par undtagelser er dog i mappen `CVRPTW/CVRPTW1`, hvor alle tre filer både findes for løsning med 30 borgere og med 50 borgere. I `CVRPTW/CVRPTW1_hold` indeholder begge mapper `Aften` og `Morgen` '.lp' og '.sol'-filer for både minimering af kørselstid og minimering af antallet af hjælpere.

