Aflevering 2: Lokalisering

Sune Lauth Gadegaard

Efterår 2022

I 2012 startede man med at bygge såkaldte supersygehuse i Danmark. Disse supersygehuse skulle centralisere driften og ikke mindst specialiseringerne inden for specifikke kerneområder. Supersygehusenes placering var under vedtagelsen et særligt ømtåleligt politisk emne, idet stor prestige og mange arbejdspladser følger med sådan et sygehuse. Derfor kan det diskuteres, om sygehusene blev placeret optimalt ud fra en objektiv målestok. Det skal undersøges i denne opgave!

Den vedlagte Excel-fil indeholder fire ark som alle er beskrevet i Tabel 1. Der er, som nævnt, i Excel-filen vedlagt data for indbyggertal for hver kommune fra forskellige årstal samt prædiktioner for fremtiden (tal fra Danmarks Statistik). Modellerne i resten af opgaven skal sørge for, at de etablerede supersygehuse har kapacitet til at håndtere de tildelte kommuners indbyggertal både nu (2022-tal) og i år 2030. Det er ikke nødvendigt, at modellen kan håndtere indbyggertallene helt frem til 2045, da det formodes, at udvidelser kan planlægges inden da.

Opgave 1: Opstil en model (det vil sige, formuler en model *matematisk*), som kan finde placeringen af 6 supersygehuse i Danmark. Modellen skal udpege 6 (forskellige) kommuner, hvor der skal etableres supersygehuse og det skal samtidig afgøres hvilke andre kommuner, der skal leverer patienter til de pågældende supersygehuse (location-allocation). Det antages, at en kommunes indbyggere tildeles helt til et supersygehus (single sourcing). Målet for modellen er at minimere de samlede omkostninger ved at etablere supersyge + omkostningen ved at etablere ekstra kapacitet ved de sygehuse hvor det er nyttigt. Implementer og løs modellen i Pyomo +Python. Visualiser og kommenter på løsningen.

Opgave 2: Det er åbenlyst ikke den smarteste måde at placere sygehuse på! Der er slet ikke taget stilling til, at flest muligt skal kunne komme hurtigt til et

Tabel 1: Oversigt over arkene i Excel filen med data

Sheet navn	Forklaring
Indbyggertal i kommuner	Indeholder data omkring indbyggertal i kommunerne i forskellige år.
Etableringsomkostninger + kap	Indeholder tre elementer: etableringsomkostninger for et "basis" supersygehus i hver kommune angivet i milliarder kroner. Kapaciteten i "basis" supersygehus målt i antal borgere et basissygehus i kommunen kan håndtere. Slutteligt angives også prisen for at udvide kapaciteten i et basissygehus med yderligere 1000 potentielle patienter. Denne pris er per 1000 ekstra patienter. Det antages, at sygehusene maksimalt kan udvides med op 50.000 ekstra potentielle patienter.
Afstande mellem kommuner (km)	Indeholder en afstandsmatrix, som angiver afstanden i kilometer mellem hvert par af kommuner (en central placering er brugt i hver kommune).
Køretid mellem kommuner (sek)	Indeholder en køretidsmatrix, som angiver hvor mange sekunder det tager, at køre fra en kommune til en anden.

supersyge; vi har blot fundet en løsning, som har lavest mulige omkostninger. I stedet bør modellen ændres, så omkostningerne ved at etablere supersygehus + ekstra kapacitet ikke overstiger omkostningen fra **Opgave 1** + 5% (denne begrænsning antages for **Opagve 2, 3, 4, 5** og **6**). Samtidig skal den samlede rejse-afstand fra kommunerne til supersygehusene minimeres. Implementer og løs modellen i Pyomo +Python. Visualiser og kommenter på løsningen.

OBS: frygt ej, hvis det tager laaaang tid at løse denne model. Den skal nok klare det! Det kan være en god idé at lave sig et lille "legetøjs" datasæt, som består af fx halvdelen af kommunerne, til at teste formuleringer på, inden man fyrer hele kanonen af på det fulde datasæt

Opgave 3: Givet, at man sjældent dør af at køre langt til et sygehus i Danmark, men oftere ved at køre *længe*, ændre da objektfunktionen til at minimere den samlede køretid. Løs den opdaterede model vha. Pyomo +Python og visualiser og kommenter på løsningen.

Opgave 4: I **Opgave 2** og **Opgave 3** blev den samlede kørselsafstand/kørselstid

minimeret uden at tage højde for, at nogle kommuner har mange indbyggere og andre få. Ændre objektfunktionen fra **Opgave 3** således, at der tages højde for antallet af indbyggere i kommunerne i 2030. Det vil sige, at kommuner med mange indbyggere skal *vægtes* højere i objektfunktionen end kommuner med få indbyggere.

Opgave 5: Indtil nu har fokus været på "effektive" løsninger, idet *summen* af kørselsafstandene er blevet minimeret. Men netop med hensyn til sygehuse kunne man argumentere for, at "retfærdighed" også er væsentligt. Ændre nu modellen til at minimere den maksimale afstand fra en kommune til den kommune hvortil den er allokeret (p-center objektfunktion).

OBS: Det kan være nødvendigt som i tilfældet med clustering af sørge for, at hvis en kommune får et sygehus, så skal kommunens egne borgere også serviceres der.

Opgave 6: Antag nu, at objektfunktionens fokus igen flyttes til den aggregerede afstand, men antag samtidig, at det er i orden at lade nogle kommuner være tilknyttet to eller flere supersygehuse (multi-sourcing). Hvorledes skal modellen ændres for at tage højde for dette? Hvilke kommuner ender med at være delt mellem to eller flere supersygehuse?

Vi skifter nu fokus fra location-allocation modeller og flytter blikket mod covering modeller. Derfor kan man nu lægger modellerne udviklet indtil nu til side.

Opgave 7: Hvis der må placeres seks supersygehuse, uden at tage stilling til prisen og kapacitetsbegrænsninger, hvor skal disse så placeres, hvis flest mulige kommuner skal kunne nå et supersygehus inden for 45 minutter? Og hvor mange kommuner kan nå et supersygehus inden for 45 minutter?

Hint: man kan her notere sig, at vi har alt hvad der skal til for at udregne en matrix (a_{ij}) som angiver om kommune j er mindre end 45 minutter fra kommune i. Man kan også notere sig, at sum-funktionen i Python kan klare det meste af arbejdet da man kan indføre en conditional i summen. Dermed bliver følgende to linjer ækvivalente

```
model.coveringCsts.add(expr=sum(model.a[i][j]*model.y[i] for i in
    model.facilityRange) >= 1)
model.coveringCsts.add(expr=sum(model.y[i] for i in model.
    facilityRange if model.a[i][j] ==1) >= 1)
```

Det kan bruges til, at forme et covering location problem uden at skulle præberegne (a_{ij})-matricen.