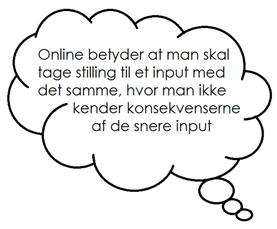
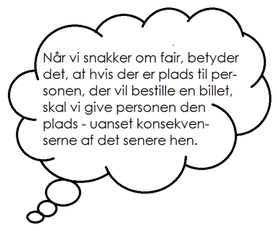


Online planlægning: Pladsreservation i tog



62a: Tenna, Martin, Nicolai, Danny og Caroline

Institut for Matematik og Datalogi,
Syddansk Universitet

Pladsreservationsproblemet

Vi har afgrænset vores projekt til at tage udgangspunkt i pladsreservationsproblemet. Det mest optimale er jo, hvis man kan få pakket et helt tog, så der aldrig var nogle tomme sæder. Men hvis uddelingen af pladser skal være *fair*, opstår der situationer, hvor du ikke kan få fyldt et sæde op på en strækning. Men hvor mange passagere er det vi mister ved det i forhold til, hvis uddelingen af pladser var *unfair*. Det har vi lavet nogle algoritmer på, hvor vi vægter lange rejser frem for flere mindre rejser, der udgør samme rejselængde. Herefter analysere vi på de dataer vi har fået ud af det, for at se om det evt. kan betale sig at skifte over til en *unfair* uddeling af pladser.

Algoritmerne

Bruteforce:

Bruteforce algoritmen er en *unfair* algoritme, som kender alle forespørgsler på forhånd. Når den så går i gang, tjekker algoritmen ALLE muligheder for hvert enkelt sæde. Muligheden, der kan give os flest passagerkilometer med færrest passager, vil altid være den mulighed vi vælger. Efter den har fundet den bedste mulighed sletter algoritmen de passagere, der var i den bedste mulighed fra passagerlisten, hvorefter vi så går i gang med næste sæde. Sådan kører det indtil vi har været igennem alle sæder.

Dynamisk:

Den dynamiske algoritme er et alternativ til bruteforce, da bruteforce er en algoritme med ekstrem køretid. Den dynamiske algoritme starter ved næstsidste station, hvor den kigger på passagerlisten om der skal nogle med dér. Er der det tilføjer vi passageren til det sæde vi er igang med at finde passagere til. Herwefter går den så et skridt tilbage, hvor den så igen kigger på om der er nogle passagere, der starter dér. Hvis der er en passager der rejser derfra og til slut vil den så tage den passager frem for to af de små, da det var vores vægtning for projektet. Sådan fortsætter den indtil den er nået til startstationen, hvor den så er færdig med et sæde. Derefter fortsætter den for de resterende sæder.

Forward-Backward:

Forward-Backward er også en *unfair* algoritme, som kender alle forespørgsler inden den går i gang. Her starter den med at kigge på alle passagerne ved startstationen, hvor den så vælger den længste. Den station, hvor passageren skal af kigger vi så i passagerlisten for at se om der er nogle, der skal på dér. Igen vægter vi den længste, hvor den fortsætter sådan indtil den når slutstationen. Bagefter gør den det samme, bare bagfra, da det kan være vi kan samle en bedre løsning ved at kigge omvendt. Men i mens den kigger omvendt har vi også nogle special cases, hvor den kigger på om der et hul mellem to passagere. Hvis der er det, kigger den for at se om der er en bedre løsning, så vi kan gøre det hul så lille som muligt. Det gør algoritmen så for enkelt sæde, indtil der ikke er flere.

Online/offline:

Online/offline algoritmen er delt op i to dele. Den starter i online delen ved at reagere på de forskellige forspørgsler én efter én. Hvis der er plads til passageren siger vi han kan få en plads, men ikke hvilken plads. Vi tænker så, at en halv time før afgang lukkes forespørgslerne ned, hvor vi så går i gang med offline delen. Her kigger den så på alle forespørglerne og finder den der starter tidligst og putter den på sædet herefter kigger den i den forspørgsels slut station om der er nogen der starter hvis der er køre den videre med den hvis ikke hopper den en station frem og kigger.

Online:

Online algoritmen er en *fair* algoritme, som tager stilling til alle forespørgsler én efter én. Den kigger først efer hvilket sæde den kan placere forspørslen på. Når den så har fundet alle sæder kigger den alle disse sæde igennem om hvor der vil komme det mindste mellemrum mellem denne forespørgsel og de tidligere forespørsler. når den har fundet det sæde med mindst muligt mellemrum placere den forspørgslen på dette sæde.

Sandsynlighed for rejser

Ud fra den måde, hvorpå vi genererer tilfældige passagerer (start- og slut-stationer), kan sandsynligheden for de forskellige rejselængder og start/slut stationer udregnes ved hjælp af formlerne nedenfor.
For at regne slut ud fra startstationen bruges denne formel:

$$\frac{1}{k} \cdot \frac{1}{i} \mid i \in \{k, k-1, \ldots, 1\}$$

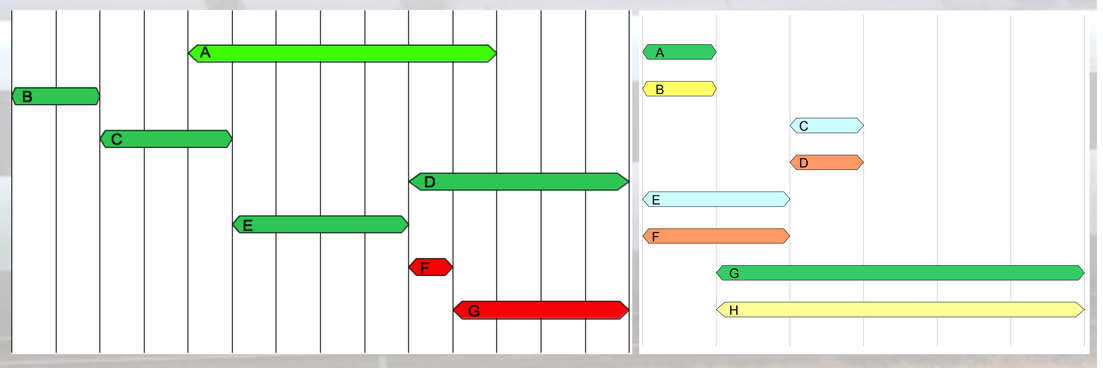
Hvis man skal overordnet lave en matematisk formel for at udregne sansynligheden for at få en hvis længde af passagerers rejse vil den se sådan ud:

$$s = \sum_{i=0}^{j-x} \frac{1}{j^2 - i \cdot j}$$

hvor x er den ønskede rejselængde og $j = k - 1$, altså antallet af stationer fratrukket en.

Hvis vi prøver med et eksempel med 3 sationer så siger første formel at vi har $\frac{1}{2}$ chance for at vi enten rammer første station. Dem som rammer første station har så $\frac{1}{2}$ chance for at stå af på anden eller tredje station. Anden formel fortæller os at der er en chance på $\frac{3}{4}$ at vi får en passager på 1 lang.

Illustrationer



Figur 1: *Begge billeder illustrere input til vores algoritmer, hvor der er 2 mulige sæder til venstre og 4 mulige sæder til højre.*

Billedet til venstre illustrerer hvordan alle vores tre offline og unfair algoritmer ville vælge. Her vil alle tre algoritmer nemlig vælge forespørgsel B, C, E og D på sæde 1, mens den vælger forespørgsel A på sæde 2, hvorefter den ikke kan få plads til forespørgsel F og G. Online/offline vælger den samme sædefordeling, men kun pga. placeringen af F og G. Havde disse forespørgsler været tidligere havde vi fået et andet resultat. Online algoritmen ville vælge A og B på sæde 1, mens den ville vælge C, D og E på sæde 2, men stadig en afvisning til F og G pga. det er de sidste.

Billedet til højre vil alle offline og unfair algoritmerne vælge at placere A og G på sæde 1, B og H på sæde 2, E og C på sæde 3, mens F og D får sæde 4. Derudover vil online/offline placere dem ligeledes, mens online vil placere A og C på sæde 1, B og D på sæde 2, E på sæde 3, F på sæde 4, hvor G og H vil blive afvist. Her kan man virkelig se hvor mange passagerkilometer, der vil gå til spilde pga. algoritmen skal være fair.

Diskussion

Kan de resultater vi har fået være teoretisk rigtigt?

NAT FF501 poster session, Juni 2015 SDU Odense