Operationsanalyse

Hans Keiding

Forord 7

Kapitel 1. Hvad er Operationsanalyse? 9
1. Indledning 9
2. Operationsanalysens historie 10
3. Operationsanalytiske problemer og metode 10
4. Litteratur 12

Kapitel 2. Lineær programmering 13
1. Indledning 13
2. Lineære programmeringsproblemer, eksempler 14
3. Lineære programmeringsproblemer, definition 15
4. Simplex-algoritmen (1): Basisløsninger 20
5. Simplex-algoritmen (2): Valg af ny basissøjle; optimalitet 24
6. Mere om dualitet 30
7. Opgaver 32
8. Litteratur 34

Kapitel 3. Mere om lineær programmering 35
1. Indledning 35
2. Dual simplex-metode 35
3. Revideret simplex 37
4. Dekomponering af LP-problemer 40
5. En anvendelse af lineær programmering: DEA-analyse 44
6. Andre LP-algoritmer 50
7. Opgaver 60
8. Litteratur 61

Kapitel 4. Ikke-lineær programmering 62
1. Indledning 62
2. Kvadratisk programmering: Wolfe's algoritme 62
3. Konveks simplex-metode 71
4. Opgaver 77
5. Litteratur 78

Kapitel 5. Heltalsprogrammering 79
1. Optimering med heltaltsrestriktioner 79
2. Metoder for heltalsprogrammering (1): Generel teori 81
3. Metoder for heltalsprogrammering (2): Gyldige uligheder 83

4. Metoder for heltalsprogrammering (3): Dualitet 86
5. Skæringsplan-algoritmer: Fractional cut 89
6. Branch-and-bound 93
7. Opgaver 97
8. Litteratur 99

Kapitel 6. Travelling Salesman 100
1. Det klassiske TSP-problem 100
2. Vehicle routing 107
3. Opgaver 111
4. Litteratur 112

Kapitel 7. Dynamisk programmering 113
1. Indledning 113
2. Diligence-eksemplet 114
3. Knapsack-problemer 116
4. Litteratur 117

Kapitel 8. Grafer 118
1. Hvad er en graf? 118
2. Nogle grafteoretiske begreber 121
3. Veje, cykler, træer 123
4. Afstand i grafer, farvning 126
5. Grader af sammenhæng i grafer 130
6. Plane grafer 131
7. Litteratur 133

Kapitel 9. Netværk 135
1. Indledning 135
2. Definition af netværk 135
3. Maximale strømme 137
4. Maximal strøm algoritmen 140
5. Edmonds-Karp algoritmen 142
6. Markeringsprocessen 144
7. Dinic's algoritme 150
8. Parring i grafer 152
9. Udvidelse af max-flow-min-cut 156
10. Opgaver 158
11. Litteratur 160

Kapitel 10. Lokalisering 161
1. Introduktion 161
2. Lokaliseringsproblemer i netværk 164
3. En algoritme til løsning af lokalisering uden kapacitetsbegrænsning 168
4. Litteratur 173

Kapitel 11. Transport og assignment 174
1. Introduktion 174
2. Teorien bag transportalgoritmen 179
3. Assignment-problemer 183
4. Opgaver 188
5. Litteratur 190

Kapitel 12. Optimale rækkefølger 191

1. Rækkefølgekrav 191
2. Projektet som netværk 191
3. Den kritiske vej 194
4. Out-of-kilter metoden til løsning af særlige LP-problemer 196
5. Anvendelse af out-of-kilter metoden til project scheduling 204
6. Flow-shop problemet 207
7. Opgaver 211
8. Litteratur 213

Kapitel 13. Kompeksitet 214
1. Indledning 214
2. Beregninger og deres kompleksitet 214
3. Decisionsproblemer 216
4. Turing maskiner 218
5. Ikke-deterministiske Turing maskiner og klassen NP 221
6. Polynomiale transformationer og NP-komplette problemer 223
7. SATISFIABILITY og Cook's sætning* 224
8. Andre NP-komplette problemer 227
9. Litteratur 229

Kapitel 14. Punktprocesser og fornyelse 230
1. Stokastiske processer i operationsanalysen 230
2. Punktprocesser 231
3. Genanskaffelsesproblemer 233
4. Fornyelsesprocesser; Blackwell's sætning 236
5. Den asymptotiske fordeling; busparadokset 241
6. Maskinproblemet med 1 operatør 243
7. Opgaver 248
8. Litteratur 249

Kapitel 15. Pålidelighed og vedligehold 250
1. Binære pålidelighedssystemer 250
2. Pålidelighedsmål 252
3. Pålidelighed i specielle situationer 255
4. Opgaver 258
5. Litteratur 258

Kapitel 16. Det optimale lager 259
1. Indledning 259
2. Det deterministiske lagerproblem 260
3. Lagerpolitik ved stokastisk efterspørgsel: Aviskioskmodellen 263
4. Stokastisk efterspørgsel: Flerperiode-modeller 265
5. Lagerpolitik ved stokastisk efterspørgsel i kontinuert tid 266
6. Poissonfordelt salg: Exakte udtryk for omkostning ved lagerpolitik 269
7. Opgaver 273
8. Litteratur 273

Kapitel 17. Køteori 274
1. Den generelle problemstilling; klassifikation af kømodeller 274
2. Little's sætning 276
3. Analyse af en kø; køens tilstand 277
4. Køen $M/M/1$ 279
5. Opgaver 283

6. Litteratur 283
Litteratur 284
Stikordsregister 287

Forord

Denne lærebog i operationsanalyse er blevet til gennem en tiårsperiode som forelæsningsnoter ved gennemgangen af faget på politstudiet ved Københavns Universitet. Ved valg af emner er der på den ene side taget hensyn til, at gennemgangen var dækkende for de problemstillinger, der sædvanligt er at finde i lærebøger i dette fag (som lineær programmering, heltalsprogrammering, strømme i netværk, kø- og lagerteori) og på den anden side har det været et mål at inddrage dels lidt nyere emner, dels sådanne, der har lidt relation til et økonomistudium iøvrigt, og som kan tjene som reklame for en matematisk tilgang til løsningen af økonomiske problemer.

Den endelige redaktion af de oprindelige forelæsningsnoter til den foreliggende form skyldes anvendelsen ved matematik-økonomi studiet på Handelshøjskolen i København.

En lang række studerende og lærere har igennem den tid, hvor materialet har været benyttet, været behjælpelige med at luge ud i de mange trykfejl; en særlig tak skal rettes til Birger Madsen, der ydede en særlig indsats i den tid, hvor han varetog undervisningen. Hvad der er tilbage, er naturligvis mit ansvar.

November 2002

Hans Keiding

KAPITEL 1

Hvad er Operationsanalyse?

1. Indledning

Der er tradition for at starte gennemgangen af et fag med en definition af dets emne. Som regel viser det sig at være temmelig svært at afgrænse emnet, for videnskabsgrene i udvikling har det med at trænge ind over de tilgrænsende discipliners område i en sådan grad, at det hurtigt bliver umuligt at adskille den ene fra den anden. Men da det nu alligevel er så rodfæstet en tradition, skal det også blive forsøgt her.

Blandt de mindst misvisende definitioner af faget operationsanalyse finder vi følgende: *Operationsanalyse er teorien for matematisk modellering af optimale beslutninger og modellernes anvendelse i praksis*. Hermed har vi ihvertfald en rimelig fornemmelse af, hvad det handler om, især når vi lægger behørig vægt på sidste del, der handler om anvendelsen i praksis, for ellers kan det være svært at se forskel på operationsanalyse på den ene side og optimering (samt store dele af nyere økonomisk teori) på den anden. Her burde der iøvrigt nok tilføjes et par ord om, hvad det er for en "praksis", der tænkes på – det er virksomheders og organisationers stræben mod bedst mulig målopfyldelse. Operationsanalysen skal ses som en erhvervsøkonomisk (i bredest tænkelige forstand) disciplin. At den også kan ses som en matematisk, er så en charme mere (eller mindre).

Der er ganske vist en del traditionelle operationsanalytiske emner (f.eks. køteori), som vi ikke får fanget ind med denne definition. Det kunne så føre til et nyt forsøg, men det kan dårligt betale sig. Operationsanalyse er simpelthen ikke nogen præcist afgrænset disciplin. Den handler om de emner, der kommer i de følgende kapitler – samt om en del andre emner, der er udeladt, herom senere. Ret meget nærmere kan man alligevel ikke komme.

Iøvrigt er disciplinens afgrænsning heller ikke konstant over tiden. Der er emner, som tidligere var oplagt operationsanalytiske (f.eks. spilteori), men som ikke er det mere, fordi de er blevet selvstændige videnskabelige discipliner og måske også er kommet til at skifte indhold og perspektiv undervejs. Til gengæld er der andre ting, der nu trænger sig mere på. Hvis man kigger på indholdsfortegnelsen, vil man se, at emner, der tidligere hørte til diskret optimering eller computer science, nu er ved at liste ind i almindelig operationsanalyse. Igen har det noget at gøre

med, at disse emner har fået anvendelser, der fører dem fra specialisternes felt over i kassen af almindeligt brugbare værktøjer.

Ser man på, hvad der normalt kommer i operationsanalytiske tidsskrifter eller på faglige konferencer, er billedet tilsvarende broget. Operationsanalyse er en slags gennemgangslejr; fremtidige videnskaber hører til her et stykke tid indtil de bliver selvstændige – eller går af mode. Det bliver den ikke mindre spændende af.

2. Operationsanalysens historie

Der er en smule uklarhed omkring det nøjagtige tidspunkt for fremkomsten af operationanalyse som en selvstændig disciplin, men det er knyttet til militær forskning – overvejende af utraditionel karakter – i England op til og under 2. verdenskrig. Udtrykket "operational research" optræder i forbindelse med udviklingen af radarsystemer i perioden 1937-39, og dette er rimeligvis første gang, at begrebet er nævnt.

For alvor skub i udviklingen kom der dog først i efterkrigstidens USA. Denne udvikling var i første omgang knyttet ret tæt til forskningen omkiring *lineær programmering,* specielt i forbindelse med simplex-metoden udviklet af G.B.Dantzig i årene efter 1945. Senere kom også andre optimeringsmetoder ind, således især diskrete optimeringsproblemer og strømme i netværk. Men gennem adskillige år var operationsanalyse 3/4 simplexmetode og 1/4 andet.

Det har ændret sig en del i de senere år. For det første er det ikke længere nødvendigt at ofre så meget opmærksomhed på simplexmetoden, da denne nu typisk vil være noget, man har lært i forvejen andetsteds. Derved bliver der automatisk mere plads til resten. Men der er for det andet også kommet en del decideret nye emner til; et eksempel på dette er multikriterieoptimering.

Da der som tidligere nævnt altid har været nogen uklarhed omkring operationsanalysens genstand, har man traditionelt understreget, at det vigtigste egentlig ikke var de konkrete emner, som man arbejder med, men *metoden,* som bliver brugt. Det vender vi tilbage til.

3. Operationsanalytiske problemer og metode

Vi har i forrige afsnit været inde på, hvilke problemstillinger, som er fast bestanddel af operationsanalysen. I den følgende oversigt har vi taget disse såvel som de senere tilkomne emner med. Operationsanalyse omfatter således følgende:

- Optimeringsproblemer:
  - Ikke-lineære optimeringsproblemer
  - Diskret optimering

- Grafmodeller:
  - Strømme i netværk
  - Lokaliseringsmodeller
  - Rækkefølger
- Spilmodeller
- Stokastiske modeller
  - Pålidelighed
  - Køteori
  - Lagerteori

Disse emner vil, pånær spilteorien, blive taget op i de følgende kapitler. Det vil ganske vist for hvert af emnerne kun blive til en begrænset fordybelse, men det er væsentligt at dække operationsanalysens mange forgreninger så godt, som det nu er muligt.

Det betyder også, at vi vil se lidt på et par emner (systemer, kompleksitet), som er lidt marginalt placeret, men som er vigtige for at forstå mange af de nye ting i branchen.

Afslutningsvis er der et par traditionelle indledende bemærkninger at fyre af. For det første plejer man at diskutere valget af *matematisk model*. Man har *ikoniske modeller,* hvor man laver en tro kopi i mindre målestok; sådanna støder vi ikke på. Videre har man *analoge* modeller, hvor man konstruerer en mekanisme fra et helt andet område, men med samme funktion som den organisation, problemet drejer sig om. Endelig har vi *symbolske* modeller, og det er typisk sådanne, vi vil få med at gøre i det følgende. Da vi kun støder på denne sidste type, er diskussionen ikke så interessant, men den understreger ihvertfald, at det i vor praktisk orienterede disciplin ikke er så vigtigt, hvordan modellen tager sig ud, det afgørende er, hvordan den fungerer.

Dermed er vi ovre i den sidste generelle kommentar, der drejer sig om *metoden*. Det er sædvanligt at se et operationsanalytisk problem som bestående af fem faser, nemlig

- Definition af problemstilling,
- Konstruktion af model,
- Løsning af model,
- Kontrol af model,
- Implementering af resultater.

Det er i det store og hele kun fase 3, som bliver behandlet i de følgende kapitler, og det har den oplagte årsag, at kun dette er egnet lærebogsstof; resten er noget, man må tilegne sig i det praktiske arbejde. Det kan dårligt undgås, at denne koncentration omkring ét af aspekterne i arbejdet med en given problemstilling må føre til en vis forsømmelse af de andre, og den dårlige samvittighed kommer til udtryk på dette sted, idet det hermed understreges, at de andre faser skam er *mindst*

lige så vigtige, uanset at de altså ikke behandles.

Der er her – som i faget generelt – en vis bevægelse. Man kan f.eks. se inddragelsen af multikriteriemetoder i operationsanalysens værktøjskasse som en begyndelse til en mere systematisk behandling af implementeringsfasen.

4. Litteratur

Der findes flere standardlærebøger i faget Operationsanalyse. De to mest udbredte er Hillier og Libermann (1995), Taha (1997). Begge er særdeles udmærkede som supplerende – eller som alternativ – læsning; de dækker nogenlunde samme emnekreds, og de kan anbefales blandt andet fordi de med et omfang på omkring 1000 sider har flere eksempler og detaljer end den foreliggende noget mindre omfangsrige fremstilling.

Et par gamle klassikere, der til gengæld ikke dækker helt så mange metoder som det kræves idag, er Churchman og Ackoff (1957) og Ackoff og Sasieni (1968).

KAPITEL 2

Lineær programmering

1. Indledning

Lineær programmering er operationsanalysens vigtigste værktøj. De fleste optimeringsopgaver vil i sidste ende blive formuleret således, at de kan løses ved metoder, der er kendte fra lineær programmering. Selv for de problemer, der ikke kan løses på denne måde, vil metoderne være anvendelige som en del af arbejdet ved at finde frem til løsninger.

Lineær programmering drejer sig egentlig blot om at maximere en lineær funktion under et antal lineære bibetingelser. At dette kan være besværligt nok, kan man dog hurtigt overbevise sig om, og der fandtes ikke systematiske løsningsmetoder før omkring midten af dette århundrede.

De første formuleringer af lineære programmeringsproblemer som sådanne, kombineret med metoder til deres løsning, kom sidst i trediverne i det daværende Sovjet, hvor de groede ud af praktiske problemer i en virksomhed, som man fik en kompetent matematiker (Kantorovich) til at se nærmere på. Gennembruddet kom under krigen med Dantzig's arbejder i USA, der også have en praktisk baggrund (forsyning af tropperne i Europa). Det er Dantzig's løsningsmetode, den såkaldte *simplex-metode,* som er blevet standard, og som vi skal bruge mest.

Betegnelsen "lineær programmering" dukkede op omkring 1950, og den har smittet af, således at andre optimeringsproblemer er blevet til "ikke-lineær programmering", "heltalsprogrammering" osv. Motivationen for denne lidt misvisende betegnelse – der er ikke tale om programmering i datalogisk forstand – var, at løsningsmetoderne i modsætning til, hvad man var vant til dengang, ikke gik på at løse et ligningssystem, men bestod i at angive en fremgangsmåde, en algoritme. Denne algoritme kan så iøvrigt føres videre og gøres til et program for en maskine, det skal vi se et par eksempler på.

Af dette kapitel – ihvertfald af de første mange afsnit – kan man godt få det indtryk, at lineær programmering er identisk med simplex-metoden. Vi skal senere løse særlige LP-problemer på andre og nemmere måder, og vi kommer også i de sidste afsnit ind på et par andre løsningsmetoder af nyere dato, så at vi samtidig kan notere os, at lineær programmering ikke er en museumsgenstand. Det gemmer vi imidlertid til et efterfølgende kapitel; i første omgang koncentrerer vi os helt om

simplex-metoden.

I det næste afsnit diskuteres nogle eksempler på lineære programmeringsproblemer, og det fører så til den generelle formulering i afsnit 3, hvorunder vi kommer ind på det meget vigtige forhold, at lineære programmeringsproblemer altid optræder i par, med et primært og et dualt problem. I afsnit 4 starter vi diskussionen af simplex-metoden, som fortsættes i afsnit 5. Derefter vender vi os til en lidt nærmere betragtning af det duale problem og hvad det kan bruges til.

2. Lineære programmeringsproblemer, eksempler

Det kommer næppe bag på nogen, at opgaver gående ud på at maximere en lineær funktion under bibetingelser givne ved lineære uligheder optræder ganske hyppigt i økonomiske problemer.

Eksempel 2.1

Den klassiske situation er virksomheden, der producerer to produkter, som sælges til faste priser, lad os sige henholdsvis 7 og 13 kroner pr. stk., og som har givne stykomkostninger, som ikke afhænger af produktionens omfang, på henholdsvis 5 og 8. Denne virksomhed forventes nu at stræbe mod maximal værdi af profitten

$$(7-5)x_1 + (13-8)x_2 = 2x_1 + 5x_2,$$

hvor produktionen af vare 1 og 2 betegnes med henholdsvis x_1 og x_2. På det korte sigt, under hvilket problemet betragtes, er der endvidere nogle kapacitetsbegrænsninger, der skyldes, at hver af produkterne lægger beslag på en vis mængde af to forskellige råvarer, af hvilke der findes et begrænset lager. Lader vi råvareforbruget pr. produceret enhed være henholdsvis 0.3 og 0.7 for den første vare, og 1.2 og 1.3 for den anden, og forudsættes de to råvarer at være til stede i mængderne 10 og 13, har vi de to bibetingelser

$$\begin{aligned} 0.3x_1 + 1.2x_2 &\leq 10 \\ 0.7x_1 + 1.3x_2 &\leq 13 \end{aligned}$$

Hertil kommer naturligvis den oplagte betingelse, at de producerede mængder ikke må være negative. Ialt har vi således et optimeringsproblem:

$$\begin{aligned} &\max\ 2x_1 + 5x_2 \\ &\text{under bibetingelserne} \\ &0.3x_1 + 1.2x_2 \leq 10 \\ &0.7x_1 + 1.3x_2 \leq 13 \\ &x_1 \geq 0, x_2 \geq 0 \end{aligned}$$

Der er en lineær funktion, der skal maximeres, der er to lineære bibetingelser, og der er ikke-negativitetsbetingelser på alle variable. Dette er så rendyrket et LP-problem, som man vel kan tænke sig.

Eksempel 2.2

Kostplan-problemer drejer sig om at sikre en forsyning $c_1, \ldots, c_m$ af forskellige næringsstoffer, der er indeholdt i diverse fødevarer, på billigste måde. Antag f.eks., at vi skal forsyne børnene i en vuggestue med kalk, protein og kulhydrater, hvoraf hver enkelt barn skal have henholdsvis 3, 5 og 4 enheder pr.dag. Vore muligheder er at indkøbe mælk, havregryn, rugbrød, Coca-Cola og Kinder-mælkesnitter. Det har vist sig, at disse varers indhold af kalk, protein og kulhydrat er som anført:

	mælk	havregryn	rugbrød	Coca-Cola	mælkesnitte
Kalk	2.3	1.2	0.3	0.1	0.2
Protein	3.1	4.2	3.1	0.0	0.3
Kulhydrat	1.1	4.3	2.2	10.5	9.1

Priserne på de fem produkter er henholdsvis 2.50, 1.50, 1.25, 6.50 og 7.50 pr.enhed; ialt har vi derfor et minimeringsproblem af formen:

$$\begin{aligned}
&\min 2.5y_1 + 1.5y_2 + 1.25y_3 + 6.5y_4 + 7.5y_5 \\
&\text{under bibetingelserne} \\
&2.3y_1 + 1.2y_2 + 0.3y_3 + 0.1y_4 + 0.2y_5 \geq 3 \\
&3.1y_1 + 4.2y_2 + 3.1y_3 + 0.3y_5 \geq 5 \\
&1.1y_1 + 4.3y_2 + 2.2y_3 + 10.5y_4 + 9.1y_5 \geq 4 \\
&y_j \geq 0, j = 1, \ldots, 5.
\end{aligned}$$

(Enhver lighed hos dette eksempel med virkelighedens verden er naturligvis tilfældig). Vi har her at gøre med et minimeringsproblem, hvor visse lineære bibetingelser ikke må overskrides. Men bortset fra det med minimeringen og retningen af ulighederne sva rer det til det første problem. At vi betegner variablene med y_j i stedet for x_i kan sådan set være ligegyldigt, men der er en vis ide også i det, som vi senere skal se.

Som det antydes af eksempelvalget, vil man normalt første gang støde på et LP-problem i forbindelse med en erhvervsøkonomisk problemstilling. Det har dog mere at gøre med, hvorledes fagene introduceres, end med anvendeligheden af LP. Det klassiske LP-problem, *kostplan-problemet* (Eksempel 2.2) der var blandt de først formulerede, er et typisk økonomisk problem om forsyning af flere typer varer.

3. Lineære programmeringsproblemer, definition

Helt generelt kan et lineært programmeringsproblem skrives som

$$\begin{aligned}
&\max c_1x_1 + \cdots + c_nx_n \\
&\text{under bibetingelserne}
\end{aligned}$$

$$\begin{aligned} a_{11}x_1 + \cdots + a_{1n}x_n &\leq b_1 \\ \vdots \qquad\qquad & \vdots \quad \vdots \\ a_{m1}x_1 + \cdots + a_{mn}x_n &\leq b_m \\ x_1 \geq 0, \ldots, x_n &\geq 0 \end{aligned} \tag{1}$$

Her er $x_1, \ldots, x_n$ de n variable i problemet. Koefficienterne $c_1, \ldots, c_m$ giver os *kriteriefunktionen*

$$z = c_1x_1 + \ldots + c_nx_n,$$

som skal maximeres. På matrixform kan det skrives

$$\begin{aligned} &\max c^t x \\ &\text{under bibetingelserne} \\ &Ax \leq b \\ &x \geq 0 \end{aligned}$$

idet tegnet ^{t} står for transponering.

Et LP-problem skrevet op som ovenfor (hvor alle bibetingelser er skrevet med ulighedstegn $\leq$) siges at være på *kanonisk form*. Det er ikke sikkert, at et problem opstået i en praktisk anvendelse ser sådan ud; Nogle af bibetingelserne kan have et ulighedstegn, som vender den anden vej ($\geq$), eller kan være givet ved $=$.

Den sidste variant er som bekendt specielt interessant, når man skal løse LP-problemer ved simplex-metoden. Vi siger at et LP-problem er på *standard form,* hvis det har følgende udseende

$$\begin{aligned} &\max c_1x_1 + \cdots + c_nx_n \\ &\text{under bibetingelserne} \\ &a_{11}x_1 + \cdots + a_{1n}x_n = b_1 \\ &\vdots \qquad\qquad \vdots \quad \vdots \\ &a_{m1}x_1 + \cdots + a_{mn}x_n = b_m \\ &x_1 \geq 0, \ldots, x_n \geq 0 \end{aligned} \tag{2}$$

eller, skrevet på matrixform,

$$\begin{aligned} &\max c^t x \\ &\text{under bibetingelserne} \\ &Ax = b \\ &x \geq 0 \end{aligned}$$

Et LP-problem på kanonisk form kan gøres til et LP-problem på standard form ved at indføre nye variable (de såkaldte slack-variable) $x_{n+1}, \ldots, x_{m+n}$, én for hver bibetingelse. At den første ulighed i (1) skal være opfyldt, er jo ensbetydende med, at

$$a_{11}x_1 + \cdots + a_{1n}x_n + x_{n+1} = b_1$$

for en vis ikke-negativ værdi af den nye variabel x_{n+1}; tilsvarende for hver af de andre uligheder. Ialt får vi den helt ækvivalente formulering af problemet:

$$\begin{aligned}
&\text{max } c_1x_1 + \cdots + c_nx_n \\
&\text{under bibetingelserne} \\
&a_{11}x_1 + \cdots + a_{1n}x_n + x_{n+1} = b_1 \\
&\quad \vdots \qquad\qquad\qquad\qquad \vdots \quad \vdots \\
&a_{m1}x_1 + \cdots + a_{mn}x_n + x_{n+m} = b_m \\
&x_1 \geq 0, \ldots, x_{n+m} \geq 0
\end{aligned}$$

Dette er tydelig nok et LP-problem på standard form; skrevet på matrixform bliver det

$$\begin{aligned}
&\text{max} \bar{c}^t \bar{x} \\
&\text{under bibetingelsen} \\
&\bar{A}\bar{x} = b \\
&\bar{x} \geq 0
\end{aligned}$$

hvor

$$\bar{A} = \begin{pmatrix}
a_{11} & a_{12} & \ldots & a_{1n} & 1 & 0 & \ldots & 0 \\
a_{21} & a_{22} & \ldots & a_{2n} & 0 & 1 & \ldots & 0 \\
\vdots & \vdots & & \vdots & \vdots & \vdots & & \vdots \\
a_{m1} & a_{m2} & \ldots & a_{mn} & 0 & 0 & \ldots & 1
\end{pmatrix}$$

med de sidste m søjler udgørende en $(m \times m)$ enhedsmatrix, hvor $\bar{c}$ er $(n+m)$-vektoren c med nuller på de sidste m pladser, og hvor $\bar{x}$ er en vektor af $n+m$ variable.

Tilsvarende kan man omskrive et LP-problem på standard form, så at det kommer på kanonisk form. For hver ligning i bibetingelserne, således f.eks. ligning i i (2), har vi, at

$$a_{i1}x_1 + \ldots + a_{in}x_n = b_i$$

er opfyldt, netop når *begge* uligheder

$$a_{i1}x_1 + \ldots + a_{in}x_n \leq b_i$$

$$a_{i1}x_1 + \ldots + a_{in}x_n \geq b_i$$

er opfyldte. Den sidste ulighed vender ganske vist forkert i forhold til kravene i den kanoniske form, men det klarer vi ved at skifte fortegn, dvs. ved at gange med -1 på begge sider, således at vi får

$$-a_{i1}x_1 - \ldots - a_{in}x_n \leq -b_i.$$

Dermed har vi omformuleret bibetingelsernes n ligninger til $2n$ uligheder, og vi har fået problemet på kanonisk form. Ganske vist har vi undervejs et fået et problem, der kan genere os lidt i sammenhæng med simplex-metoden, nemlig at nogle af koefficienterne b_i kan være negative, men det kan vi klare hen ad vejen.

Vi skal bruge lidt terminologi: En vektor x, der opfylder bibetingelserne, kaldes en *brugbar løsning;* hvis den maximerer kriteriefunktionen blandt alle brugbare løsninger, kaldes den en *optimal løsning*.

Det er ikke sikkert, at der overhovedet findes brugbare løsninger; det gør der f.eks. ikke, hvis bibetingelserne strider mod hinanden, som i tilfældet

$$\begin{aligned} x_1 + x_2 &\leq 1 \\ -x_1 + x_2 &\leq -2 \end{aligned}$$

hvor det ses, at x_2 må være negativ, for at begge uligheder kan være opfyldte.

Selvom der findes brugbare løsninger, behøver der ikke at være optimale; det hænger sammen med, at problemet kan være *ubegrænset,* som det er tilfældet i følgende eksempel:

$$\begin{aligned} &\max x_1 + x_2 \\ &\text{under bibetingelserne} \\ &x_1 - 2x_2 = 17 \\ &x_1 \geq 0, x_2 \geq 0 \end{aligned}$$

Her vil vi kunne opnå vilkårligt store kriterieværdier: Når blot

$$x_2 = \frac{x_1 - 17}{2},$$

kan x_1 vælges vilkårligt stor uden at bibetingelserne er overtrådt, og kriteriefunktionens værdi bliver da

$$x_1 + \frac{x_1 - 17}{2} = \frac{3}{2}x_1 - \frac{17}{2},$$

der klart nok er ubegrænset i x_1.

Dualitet. Lineære programmeringsproblemer optræder så at sige altid i par; til et LP-problem (i denne forbindelse kaldt "primært")

$$\begin{aligned}
&\max c_1x_1 + \cdots + c_nx_n \\
&\text{under bibetingelserne} \\
&a_{11}x_1 + \cdots + a_{1n}x_n \leq b_1 \\
&\quad\vdots \qquad\qquad\quad \vdots \qquad \vdots \\
&a_{m1}x_1 + \cdots + a_{mn}x_n \leq b_m \\
&x_1 \geq 0, \ldots, x_n \geq 0,
\end{aligned}$$

hører der et *dualt* LP-problem, nemlig

$$\begin{aligned}
&\min b_1y_1 + \cdots + b_my_m \\
&\text{under bibetingelserne} \\
&a_{11}y_1 + \cdots + a_{m1}y_m \geq c_1 \\
&\quad\vdots \qquad\qquad\quad \vdots \qquad \vdots \\
&a_{1n}y_1 + \cdots + a_{mn}y_m \geq c_n \\
&y_1 \geq 0, \ldots, y_m \geq 0.
\end{aligned}$$

Der er indført nye variable $y_1, \ldots, y_m$, og der minimeres, men iøvrigt er koefficienterne, såvel i kriteriefunktion som i bibetingelser, de oprindelige, blot er der ordnet om på dem. Sammenhængen kommer klart frem, hvis henholdsvis primært og dualt LP-problem skrives på matrixform,

Primært problem:	Dualt problem
$\max c^tx$	$\min b^ty$
under bibetingelserne	under bibetingelserne
$Ax \leq b$	$y^tA \geq c^t$
$x \geq 0$	$y \geq 0$

Det er let at se, at hvis x er en brugbar løsning til det primære problem og y en brugbar løsning til det duale, da må der gælde, at

$$c^tx \leq (y^tA)x = y^t(Ax) \leq y^tb = b^ty,$$

så kriteriefunktionen i det primære problem altid er $\leq$ værdien af kriteriefunktionen i det duale.

Der gælder mere endnu, nemlig den såkaldte hovedsætning om lineær programmering, som vi faktisk vil få et bevis for i forbindelse med vor diskussion af

simplex-algoritmen: Hvis både det primære og det duale problem har en brugbar løsning, da har de begge en optimal løsning, og for disse er værdien af kriteriefunktionerne ens. Hvis ét af problemerne ikke har en brugbar løsning, da har ingen af dem en optimal løsning.

4. Simplex-algoritmen (1): Basisløsninger

I det generelle tilfælde med n variable har vi brug for en metode, der giver os mulighed for systematisk at flytte fra en brugbar basisløsning til en anden, således at kriterieværdien forbedres undervejs. En sådan metode er *simplex-metoden,* som vi vil forklare i dette og næste afsnit.

Udgangspunktet er et LP-problem på *standard form*,

$$\begin{aligned} &\max c^t x \\ &\text{under bibetingelserne} \\ &Ax = b \\ &x \geq 0 \end{aligned}$$

Det antages, at alle b_i er ikke-negative (i modsat fald kan den pågældende ligning ganges igennem med -1 uden at det ændrer noget ved mængden af løsninger).

Vi skal i første omgang kun interessere os for *systemet af bibetingelser*

$$Ax = b,\ x \geq 0.$$

Dette er næsten et almindeligt problem om løsning af et antal lineære ligninger, noget som kan gøres f.eks. ved at løse en enkelt ligning og substituere i den næste osv. Vi skal dog også holde styr på, at vi hele tiden kun har *ikke-negative* løsninger.

I alt det følgende antages det, at $m \leq n$ og at der er m lineært uafhængige rækker i matricen A. Det betyder, at der ikke er nogen række, der kan fås som en linearkombination af de øvrige. Var der det, kunne vi blot smide den væk.

Vi indfører betegnelsen en *basis* for en samling af m lineært uafhængige søjler fra A. Skriver vi j'te søjle som A_j, vil en basis altså kunne skrives som

$$\mathcal{B} = \{A_{j_1}, \ldots, A_{j_m}\},$$

hvor der er udtaget søjler med index $j_1, \ldots, j_m$. En *basisløsning* hørende til $\mathcal{B}$ er en løsning x til ligningssystemet $Ax = b$, således at $x_j = 0$ for $j \notin \{j_1, \ldots, j_m\}$ (koefficienter hørende til ikke-basissøjler er nul). En *brugbar basisløsning* hørende til B er en basisløsning med ikke-negative koefficienter.

Skrives $m \times m$ matricen af basissøjler fra $\mathcal{B}$ som B, og er x en basisløsning, har vi

$$Bx = b \text{ eller } x = B^{-1}b.$$

De to ligninger kan også skrives som

$$b = \sum_{j=1}^{m} x_j B_j, \quad x = \sum_{j=1}^{m} b_j (B^{-1})_j,$$

hvor B_j, henholdsvis $(B^{-1})_j$, er den j'te søjle i B, henholdsvis B^{-1}, den inverse matrix til B. Simplex-metoden går i al korthed ud på at flytte sig fra en brugbar basisløsning til en anden, således at kriteriefunktionens værdi hele tiden vokser. Når man ikke kan øge kriteriefunktionen ved at flytte til en anden brugbar basisløsning, har man et optimum.

For at udføre disse ting i praksis må vi have en bekvem regnemetode. Den fås af det såkaldte *simplex-tableau*. Tableauet har $m+1$ rækker og $n+1$ søjler; der er en søjle (den første) svarende til vektoren b samt en søjle for hver variabel. Tilsvarende er der en række (den øverste) til vektoren c og en række til hver bibetingelse. Det øverste højre hjørne i tableauet står tomt i første omgang:

	c_1	$\cdots$	c_n
b_1	a_{11}	$\cdots$	a_{1n}
$\vdots$	$\vdots$		$\vdots$
b_m	a_{m1}	$\cdots$	a_{mn}

Søjlen af b-koefficienter må ikke indeholde negative tal; ellers er der ingen betingelser på koefficienterne. Vi vil i første kun se på de nederste m rækker, så vi udelader rækken med koefficienter fra vektoren c. Lad os i starten antage, at vort tableau er således, at *de sidste m søjler udgør en enhedsmatrix*, dvs. at vi har tableauet

b_1	a_{11}	$\cdots$	$a_{1,n-m}$	1	$\cdots$	0
$\vdots$	$\vdots$		$\vdots$	$\vdots$		$\vdots$
b_m	a_{m1}	$\cdots$	$a_{m,n-m}$	0	$\cdots$	1

Hvis de sidste m variable var slack-variable af den type, vi brugte for at bringe problemet fra kanonisk til standard form, vil tableauet se ud netop på denne måde. Vi ser, at de sidste m søjler netop giver os en basis – vi har at b kan skrives som en ikke-negativ linearkombination af søjlerne $A_{n-m+1}, \ldots, A_n$, nemlig på den trivielle måde

$$b = b_1 A_{n-m+1} + \cdots + b_m A_n.$$

Eksempel 2.3

Betragt det lineære programmeringsproblem

$$\begin{aligned}
&\max 2x_1 + 2x_2 + x_3 \\
&\text{under bibetingelserne} \\
&x_1 + x_2 + 2x_3 \leq 2 \\
&4x_1 + 2x_2 + x_3 \leq 4 \\
&x_1 + 2x_2 + x_3 \leq 2 \\
&x_1, x_2, x_3 \geq 0.
\end{aligned}$$

For at kunne løse problemet ved simplex-metoden skal vi først omskrive til standard form, idet vi indfører nye slack-variable x_4, x_5, x_6 svarende til hver ulighed, hvorefter vort problem ser således ud:

$$\begin{aligned}
&\max 2x_1 + 2x_2 + x_3 \\
&\text{under bibetingelserne} \\
&x_1 + x_2 + 2x_3 + x_4 = 2 \\
&4x_1 + 2x_2 + x_3 + x_5 = 4 \\
&x_1 + 2x_2 + x_3 + x_6 = 2 \\
&x_1, x_2, x_3, x_4, x_5, x_6 \geq 0.
\end{aligned}$$

Opstillet i tableau-form får vi følgende:

	2	2	1	0	0	0
2	1	1	2	1	0	0
4	4	2	1	0	1	0
2	1	2	1	0	0	1

Bemærk, at de tre sidste søjler udgør en enhedsmatrix. Det vil altid være tilfældet, når der er føjet slack-variable til problemet svarende til samtlige bibetingelser.

Basisløsningen er altså $(x_1, \ldots, x_m) = (b_1, \ldots, b_m)$, så yderste venstre søjle angiver løsningen: Tallet i række i er værdien af den variabel fra basis, hvis tilhørende søjle har ettallet i i'te række. Dette er et gennemgående fænomen ved simplex-tableauet; man kan altid aflæse den aktuelle basisløsning.

Der kan læses mere endnu: Tager vi en søjle A_j, som *ikke* er i basis, har vi, at dens koefficienter er a_{ij} er de tal, som basissøjlerne skal ganges med for at få j'te søjle frem. Igen er det naturligvis helt trivielt for vor starttableau, men det er det ikke, når vi begynder at ændre på det.

Ændringer i tableauet sker på en eneste måde, nemlig ved *rækkeoperationer.* Man må gange en vilkårlig række med et tal (positivt eller negativt) og lægge til en vilkårlig anden række. Hvis dette er gjort således, at det tableau, som kommer ud af det, har en basis i form af enhedsvektorer stående i passende søjler $\{j_1, \ldots, j_m\}$ (således at disse giver et ettal i hver af rækkerne, ikke nødvendigvis i nummerorden),

kan vi stadig aflæse basisløsning af yderste venstre søjle, og inde i tableauet kan vi aflæse de oprindelige søjlers koefficienter i den nye basis.

I det følgende vil vi overalt bruge notation b, A, B osv. for vektorer og matricer lavet af koefficienterne fra *det oprindelige problem* angivet ovenfor. Vi bruger samme bogstaver, men med en ˆ føjet på, til at angive koefficienter i et simplex-tableau lavet fra det oprindelige ved hjælp af passende mange rækkeoperationer. I hvert sådant tableau har vi, når B er matricen hørende til basis, at yderste venstre søjle er den aktuelle basisløsning,

$$b = \sum_{j=1}^{m} \hat{b}_j B_j,$$

(idet $\hat{x}_j = 0$ hvis j ikke er en søjle fra den aktuelle basis, og j'te søjle i tableauet er den oprindelige søjle j's koordinater i den aktuelle basis:

$$A_j = \sum_{i=1}^{m} \hat{a}_{ij} B_i.$$

Disse sammenhænge følger af, at vi har brugt rækkeoperationer til at få en basis frem. Vi skal bruge dem i det følgende.

Hermed har vi værktøjet til at forklare simplex-metoden: Lad os antage, at vort tableau efter diverse rækkeoperationer er følgende:

$\hat{b}_1$	$\hat{a}_{11}$	$\cdots$	$\hat{a}_{1n}$
$\vdots$	$\vdots$		$\vdots$
$\hat{b}_m$	$\hat{a}_{m1}$	$\cdots$	$\hat{a}_{mn}$

og at der er en basis svarende til søjlerne $\{j_1, \ldots, j_m\}$. Den tilhørende basisløsning $\hat{x}$ antages at have alle koordinater svarende til basis strengt positive.

Vi ønsker at ændre på vor basis $\mathcal{B}$ ved at inddrage en ny søjle A_j. I så fald må vi droppe en af de eksisterende søjler, for der kan ikke være flere end m, hvis søjlerne skal være lineært uafhængige. Vi kan dog ikke fjerne en søjle fra den gamle basis på vilkårlig måde; der skal tages hensyn til, at vi skal kunne skrive b som en ikke-negativ linearkombination af de nye basissøjler.

Vi har som bemærket ovenfor, at

$$b = \hat{b}_1 A_{j_1} + \cdots + \hat{b}_m A_{j_m} \tag{1}$$

med $\hat{b}_i > 0,\ i = 1, \ldots, m$. Videre har vi, at enhver søjle A_j kan skrives som en linearkombination af basissøjlerne, nemlig som

$$A_j = \hat{a}_{1j} A_{j_1} + \cdots + \hat{a}_{mj} A_{j_m}$$

eller

$$0 = \lambda A_j - \lambda \sum_{i=1}^{m} \hat{a}_{ij} A_{j_i}, \tag{2}$$

hvor vi har flyttet rundt på leddene og ganget på begge sider med et (positivt) tal λ. Vi ved ikke noget om fortegnene på koefficienterne $\hat{a}_{1j}, \ldots, \hat{a}_{mj}$, men hvis vi lægger (1) og (2) sammen, får vi, at

$$b = \sum_{i=1}^{m} (\hat{b}_i - \lambda \hat{a}_{ij}) A_{j_i} + \lambda A_j,$$

og dette udtryk fortæller os, at når tallet λ er > 0 og meget lille, så vil alle koefficienter være ikke-negative. Vælger vi tilmed λ præcist så stort, at netop én af koefficienterne til de gamle basissøjler bliver 0, får vi en ny basis med A_j inde og en af de gamle ude. Den præcise værdi af λ er

$$\lambda = \min_{j:\hat{a}_{ij}>0} \frac{\hat{b}_i}{\hat{a}_{ij}}.$$

I simplex-tableauet svarer det til, at vi række for række danner forholdet mellem koefficienterne i yderste venstre søjle og i den potentielle basissøjle, med mindre den sidste er 0 eller negativ. Vi vælger da den række, for hvilket det resulterende tal er mindst (er der mere en én sådan række, vælges den øverste af dem). Vi ved nu, hvilken basissøjle som skal ud, nemlig den, der har ettallet i den pågældende række.

Dette klarer basisskift i den situation, hvor den aktuelle basisløsning $\hat{b}$ *ikke* har nogen koefficienter, som er nul. Hvis der er en sådan, f.eks. $b_{i_0} = 0$ kan vi ikke bruge argumentet ovenfor, men vi kan da blot lade den nye søjle gå ind i basis i stedet for den i_0'te, igen med den tilhørende koefficient lig nul. Der er sådan set ikke sket nogetsomhelst, men det er praktisk alligevel at betragte dette som et basisskift, og i tableauet fungerer det helt på samme måde.

Hele operationen kaldes et *pivotstep,* og man siger, at vi *pivoterer* omkring det særlige element i tableauet valgt ovenfor.

5. Simplex-algoritmen (2): Valg af ny basissøjle; optimalitet

Vi ved nu, hvorledes vi flytter fra en basis til en anden. Der resterer at forklare, efter hvilket kriterie vi udvælger den nye søjle til basis.

Her skal vi bruge den første række i simplex-tableauet. Vi vil igen antage, at vort første simplex-tableau er sådan, at de sidste $n - m$ variable er slackvariable; da de ikke indgik i det oprindelige LP-problem, er de tilhørende koefficienter i

Eksempel 2.4

Antag, at vi i tableauet fra sidste eksempel, som her er gengivet uden øverste række,

2	1	1	2	1	0	0
4	4	2	1	0	1	0
2	1	2	1	0	0	1

ønsker at skifte den første søjle ind i basis. Vi sammenligner da tallene i b-søjlen divideret samme rækkes tal i den betragtede søjle; det giver os henholdsvis

$$\frac{2}{1} = 2,\ \frac{4}{4} = 1 \text{ og } \frac{2}{1} = 2.$$

Det mindste af disse tal er 1, så vi skal pivotere om elementet i første søjle, anden række. (Bemærk, at alle tal i den betragtede søjle var positive; hvis der havde været negative tal eller 0, skulle de pågældende rækker ikke indgå i sammenligningen.)

Først divideres der igennem i anden række med pivotelementet 4. Det giver os en række

1	1	$\frac{1}{2}$	$\frac{1}{4}$	0	$\frac{1}{4}$	0

med et ettal på pivotelementets plads. For at få nul i første søjle på de øvrige pladser kan vi trække den nye anden række fra henholdsvis første og tredie række. Ialt giver det os tableauet

1	0	$\frac{1}{2}$	$\frac{7}{4}$	1	$-\frac{1}{4}$	0
1	1	$\frac{1}{2}$	$\frac{1}{4}$	0	$\frac{1}{4}$	0
1	0	$\frac{3}{2}$	$\frac{3}{4}$	0	$-\frac{1}{4}$	1

Ialt er der, som man kan se, sket det, at første søjle er kommet ind i basis i stedet for den femte søjle. At det var femte søjle, der skulle ud af basis, fandt vi faktisk ud af, allerede da vi fandt pivotelementet. Det stod nemlig i anden række, og hvis vi gik hen til den basisvektor, der havde sit ettal i denne række, ville vi netop være kommet til række fem.

Ved vort basisskift har vi fået en ny brugbar basisløsning frem i b-søjlen. Der står nemlig løsningen $x_4 = 1, x_1 = 1, x_6 = 1$ (og resten lig nul).

vektoren c alle nul:

	c_1	$\cdots$	c_{n-m}	0	$\cdots$	0
b_1	a_{11}	$\cdots$	$a_{1,n-m}$	1	$\cdots$	0
$\vdots$	$\vdots$		$\vdots$	$\vdots$		$\vdots$
b_m	a_{m1}	$\cdots$	$a_{m,n-m}$	0	$\cdots$	1

Som tidligere bruger vi ^-notationen for koefficienter i de efterfølgende tableauer, hvor der er udført rækkeoperationer. Bemærk, at der dermed også vil komme tal i øverste venstre hjørne. Dette tal skriver vi som $-z$ (idet z selv bliver ikke-negativ).

At inddrage den øverste række ændrer ikke noget ved operationerne omkring basisskift. Det eneste, der skal føjes til, er at man ved hvert basisskift skal sørge for, at *$\hat{c}$-koefficienterne i basissøjlerne er 0*.

Antag nu, at vi efter et antal basisskift har tableauet

	$\hat{c}_1$	$\cdots$	$\hat{c}_n$
$\hat{b}_1$	$\hat{a}_{11}$	$\cdots$	$\hat{a}_{1n}$
$\vdots$	$\vdots$		$\vdots$
$\hat{b}_m$	$\hat{a}_{m1}$	$\cdots$	$\hat{a}_{mn}$

Ved forskriften om nul i øverste række af basissøjler opnår vi, at

$$\hat{c}_j = c_j - \sum_{i=1}^{m} c_i \hat{a}_{ij},$$

hvor der summeres over den aktuelle basis. Vi noterer os nu, at $\hat{a}_{ij}$ for $i = 1, \ldots, m$ er den j'te søjles koordinater i den aktuelle basis. Vi kan derfor fortolke udtrykket ovenfor: c_j er gevinsten (målt i kriteriefunktionen) ved at bringe én enhed af søjle j ind i basis; andet led er tabet (igen målt i kriteriefunktion) ved at lade denne enhed gå ind, for vi må jo så til gengæld lade tilsvarende meget af den gamle basis gå ud. Det er nu oplagt, at vi alt i alt får en gevinst ved at bringe søjle j ind, når $\hat{c}_j > 0$, ellers ikke.

Det fører os til vort kriterie for at bringe en ny søjle ind i basis: Vælg blandt alle de søjler udenfor basis, for hvilke $\hat{c}_j > 0$, den søjle med størst værdi af $\hat{c}_j$ (er der flere af dem, da vælg den længst til venstre i tableauet). *Hvis der ikke er nogen søjle med $\hat{c}_j > 0$, er vi færdige*.

Vi får iøvrigt den optimale værdi af kriteriefunktionen serveret i tableauet: I øverste venstre hjørne har vi nemlig

$$-z = -\sum_{i=1}^{m} \hat{b}_i c_i$$

og da $\hat{b}$ netop er ikke-nul-elementerne i løsningen $\hat{x}$, får vi z som den aktuelle kriterieværdi. Er vi nået til sidste tableau, hvor vi ikke kan komme videre, er $\hat{x}$ den optimale løsning, z den optimale kriterieværdi.

Der er en lidt teknisk pointe i forbindelse med simplex-algoritmen: Vi kan se fra konstruktionen i det foregående, at kriteriefunktionens værdi aldrig falder i forbindelse med et basisskift. Den kan dog forblive uændret, noget som sker hvis man fra en situation, hvor der var nuller i basisløsningen, skifter over til en anden basisløsning med nuller. Dette sker der ikke noget ved i sig selv, men selve det forhold, at kriteriefunktionen ikke vokser ved hvert basisskift, åbner en teoretisk mulighed for, at man kan skifte basis en del gange og derefter være kommet tilbage til en basis, men allerede har haft. Det ville betyde, at algoritmen har *cykler,* går i ring, en ganske ubehagelig egenskab, som man helst vil være foruden.

Hvad angår simplex-algoritmen, så kan den slags undgås ved en passende specifikation af, hvad man skal gøre i situationer, hvor der er mere end én mulighed for at vælge (søjle eller række). Denne specifikation kan ske på forskellig måde; den, som er valgt i teksten, kaldes Bland's anticykling-algoritme.

Vi kan nu sammenfatte simplexalgoritmen i følgende trin:

> **1. Klargøring:** Opstil det første simplex-tableau. Sørg for, at b-søjlen er ikke-negativ, at der er en basis svarende til enhedsvektorer, og at øverste række har 0 i basissøjlerne.
>
> **2. Inddrage ny søjle:** Så længe der er en søjle, så den tilhørende koefficient er positiv, gennemføres dette og næste trin: Vælg den søjle j, som har størst koefficient i første række (hvis der er flere, vælg den som står længst mod venstre).
>
> **3. Pivotstep:** Vælg den række i, for hvilken forholdet $\hat{b}_i/\hat{a}_{ij}$ er mindst (idet der kun ses på rækker, hvor $\hat{a}_{ij} > 0$). Lad søjle i erstatte søjle j i basis, dvs. lav rækkeoperationer således at ij'te element bliver 1, og alle andre elementer i j'te søjle (inclusive elementet i øverste række) bliver 0.

Dermed har vi alle elementer i simplex-algoritmen pånær et enkelt: Hvad gør man, hvis problemet i sin oprindelige udgave er sådan, at der *ikke* skal tilføjes slackvariable, og man *ikke* umiddelbart har en basis i sit starttableau?

At dette er et reelt problem, bliver især klart når problemets dimension er passende stort; for problemer med to-tre rækker kan man nok finde en basis ved at prøve sig frem, men generelt må man have en fast procedure.

Eksempel 2.5

Lad os regne vort gennemgående eksempel helt igennem ved hjælp af simplex-metoden. Starttableauet var

	2	2	1	0	0	0
2	1	1	2	1	0	0
4	4	2	1	0	1	0
2	1	2	1	0	0	1

Tableauet er allerede klargjort, idet der er en basis (de sidste tre søjler), og koefficienterne i c-rækken hørende til basissøjler er alle 0.

Vi ser, at både første og anden søjle har maximal koefficient i c-rækken, nemlig 2. Vi vælger da den med lavest nummer, dvs. søjle 1.

Vi har i tidligere set, hvordan vi finder pivotelement og tilpasser tableauet; der skal her blot også skaffes et 0 i c-rækken, hvilket sker ved at gange den nye anden række med 2 og trække fra c-rækken. Resultatet bliver

-2	0	1	$\frac{1}{2}$	0	$-\frac{1}{2}$	0
1	0	$\frac{1}{2}$	$\frac{7}{4}$	1	$-\frac{1}{4}$	0
1	1	$\frac{1}{2}$	$\frac{1}{4}$	0	$\frac{1}{4}$	0
1	0	$\frac{3}{2}$	$\frac{3}{4}$	0	$-\frac{1}{4}$	1

I det nye tableau har anden søjle den største positive c-koefficient og skal altså ind i basis. Der pivoteres om elementet i tredie række, og vi får tableauet

$-\frac{8}{3}$	0	0	0	0	$-\frac{1}{3}$	$-\frac{2}{3}$
$\frac{2}{3}$	0	0	$\frac{3}{2}$	1	$-\frac{1}{6}$	$-\frac{1}{3}$
$\frac{2}{3}$	1	0	0	0	$\frac{1}{3}$	$-\frac{1}{3}$
$\frac{2}{3}$	0	1	$\frac{1}{2}$	0	$-\frac{1}{6}$	$\frac{2}{3}$

I dette tableau har c-rækken udelukkende ikke-positive elementer. Det betyder, at vi har nået den optimale løsning. Denne kan aflæses i b-søjlen til

$$x_1 = \frac{2}{3},\ x_2 = \frac{2}{3},\ x_4 = \frac{2}{3},$$

Eksempel 2.5, fortsat

og de øvrige variable lig nul. Da det kun er de første tre variable, der er af interesse, kan vi altså konkludere, at løsningen til det oprindelige problem i eksempel 7.1 bliver

$$x_1 = \frac{2}{3},\ x_2 = \frac{2}{3},\ x_3 = 0.$$

Man gør følgende: Antag, at det oprindelige problem er på standard form. Tilføj m nye slackvariable $x'_{n+1}, \ldots, x'_{n+m}$, og opstil *hjælpeproblemet*

$$\begin{aligned} &\max -x'_{n+1} - \cdots - x'_{n+m} \\ &\text{under bibetingelserne} \\ &Ax + Ix' = b \\ &x \geq 0, x' \geq 0. \end{aligned}$$

Dette er et nyt LP-problem på standard form. Det er uheldigvis lidt større end det oprindelige, men det har til gengæld den fordel, at vi umiddelbart kan nedskrive et simplex-tableau med en basis i de sidste m søjler.

Der køres nu simplex på hjælpeproblemet, hvorved der kan ske tre forskellige ting:

(1) På et tidspunkt har vi en basis af søjler svarende til de "gamle" variable, således at alle slackvariablene er drevet ud af basis. Da har vi klart nok også en basis i det oprindelige problem. Da det var det, vi manglede, kan vi nu gå i gang med det gamle problem.

(2) Den optimale løsning til hjælpeproblemet indeholder slackvariable på ikke-nul niveau, svarende til at kriterieværdien i maximum er negativ. Da vil der ikke være nogen brugbar basisløsning til det oprindelige problem, for i så fald ville vi have en optimal løsning i vort hjælpeproblem med kriterieværdi 0.

(3) Vi får en optimal løsning til hjælpeproblemet med $z = 0$, men nogle af slackvariablene er i basis på nul-niveau. I denne, lidt specielle situation har vi en brugbar løsning til det oprindelige problem, men vi mangler nogle søjler for at have en basis. Her kan man komme på plads ved at inddrage vilkårlige søjler og bytte ud mod slackvariabel-søjlerne (der skal blot være noget forskellig fra 0 på ij'te plads, hvis søjle j skal ind i stedet for basissøjle i).

Minimering. Til afslutning vender vi tilbage til problemet om *minimering* i stedet for maximering. Som vi tidligere har set, kan minimering af funktionen f gennemføres som maximering af $-f$. Oversat til simplex-tableau betyder det, at vi blot skal starte med et fortegnsskift på alle c-koefficienter og derefter gennemføre simplex ganske som tidligere. Løsningen vil da fremkomme som sædvanlig; eneste forskel fra tidligere er, at vi tilsvarende skal vende fortegn på den optimale kriterieværdi, hvilket iøvrigt betyder, at den kommer direkte frem i øverste venstre hjørne.

Alternativt kan man også operere med en særlig simplex-rutine for minimeringsproblemer. I så fald skrives de rigtige c-koefficienter i øverste række, men ved afgørelsen af, hvilken søjle som skal ind i basis, er det nu den med mindst mulig koefficient, der skal hentes ind, og vi har en optimal løsning når alle koefficienter er positive. Alt andet er som ved simplex for maximeringsproblemer. Denne sidste procedure er iøvrigt helt ækvivalent med den ovenfor, så det er en smags sag, hvad man vælger.

6. Mere om dualitet

Simplex-tableauet giver os endnu en information (foruden optimal basisløsning og optimal kriterieværdi), nemlig løsningen til *det duale problem*.

Til dette formål er det vigtigt at starte med et klargjort tableau

	c_1	$\cdots$	c_{n-m}	0	$\cdots$	0
b_1	a_{11}	$\cdots$	$a_{1,n-m}$	1	$\cdots$	0
$\vdots$	$\vdots$		$\vdots$	$\vdots$		$\vdots$
b_m	a_{m1}	$\cdots$	$a_{m,n-m}$	0	$\cdots$	1

Når der gennemføres en hel række af pivotstep, vil tableauet få nye koefficienter; specielt vil der i øverste række typisk ikke længere stå 0 på de sidste m pladser, men passende tal $\hat{c}_{n-m+1}, \ldots, \hat{c}_n$.

Det første, vi kan notere os, er at vektoren $\hat{c}_S$ af koefficienter $\hat{c}_{n-m+j}$ hørende til slackvariablene kan skrives som

$$\hat{c}_S = -c_B^t B^{-1},$$

hvor c_B er vektoren af oprindelige c-koefficienter til den aktuelle basis, og B er matricen af søjler hørende til denne basis. Vi får heraf, at

$$\sum_{i=1}^{m} \hat{c}_{n-m+i} b_i = \hat{c}_S^t b = -c_B^t B^{-1} b = -c^t \hat{x},$$

hvor $\hat{x}$ er den aktuelle brugbare basisløsning.

Som man kan se, er der her en mulighed for at gøre prøve, idet vi åbenbart skal have, at $\sum_{i=1}^{m} \hat{c}_{n-m+i} b_i = -z$. Værdien af denne prøve skal dog ikke overvurderes; den fanger ikke alle tænkelige regnefejl, og det er da heller ikke derfor, at c-koefficienterne hørende til slackvariablene er interessante.

Pointen er, at når vi har den optimale løsning, er $-\hat{c}_{n-m+i}$ for $i = 1, \ldots, m$ *den optimale løsning til det duale problem*. Disse tal er for det første ikke-negative; ellers ville vi jo ikke være i optimum. De giver også en brugbar løsning til det duale problem, for hvis vi tager j'te søjle i A, har vi

$$(-\hat{c}_S^t)A_j = c_B^t B^{-1} A_j.$$

Nu er sidste led netop det, der skal trækkes fra c_j for at få $\hat{c}_j$. Men da vi var i optimum, er alle $\hat{c}_j$ ikke-positive, hvilket betyder, at der må gælde

$$c_B^t B^{-1} A_j \geq c_j$$

for alle j, dvs. $-\hat{c}_S^t$ er en brugbar løsning til det duale problem.

Det er også en optimal løsning: Lad nemlig $y \in \mathbb{R}_+^m$ med $y \geq 0$ være således at

$$y^t A \geq c^t;$$

da dette er en ulighed, som holder for hver af de n koordinater, må den specielt holde for dem, der hører til den aktuelle basis, og vi kan skrive dette som

$$y^t B \geq c_B^t.$$

Vi bruger nu, at vi har en optimal løsning $\hat{x}$ med ikke-nulelementer svarende til vektoren $\hat{b}$; ganges hver af de m uligheder med $\hat{b}_i \geq 0$, $i = 1, \ldots, m$, og lægges sammen, fås – idet der benyttes at $\hat{b} = B^{-1}b$ – at

$$y^t B B^{-1} b \geq c_B^t \hat{b},$$

der også kan skrives som

$$y^t b \geq c^t \hat{x} = -\hat{c}_S^t b,$$

hvoraf vi ser, at enhver brugbar løsning til det duale problem må have mindst lige så stor kriterieværdi som vektoren $-\hat{c}_S$.

Denne egenskab ved simplex-metoden, at den giver de duale variable som et automatisk biprodukt, er særdeles nyttig. Dels kan det ofte være netop disse variable, man er interesseret i, dels har de tit en særlig økonomisk fortolkning, som kaster nyt lys på det oprindelige problem. Det er derfor praktisk, at de kan aflæses umiddelbart.

Ser man nærmere til, vil man af det foregående faktisk kunne udlede et bevis for den tidligere omtalte hovedsætning for lineær programmering. Vi har ihvertfald vist, at hvis det primære problem har optimal løsning, da kan en sådan findes ved simplex-metoden, som samtidig giver os en optimal løsning til det duale problem, endda med samme kriterieværdi. Resten af sætningen kan fås ved at flytte lidt om på problemet; det overlades til læseren.

Eksempel 2.6

I det gennemgående eksempel kan vi finde løsningen til det duale problem

$$\begin{aligned}&\min 2y_1 + 4y_2 + 2y_3\\ &\text{under bibetingelserne}\\ &y_1 + 4y_2 + y_3 \geq 2\\ &y_1 + 2y_2 + 2y_3 \geq 2\\ &2y_1 + y_2 + y_3 \geq 1\\ &y_1, y_2, y_3 \geq 0.\end{aligned}$$

ved at aflæse koefficienterne til slack-variablene:

$-\frac{8}{3}$	0	0	0	0	$-\frac{1}{3}$	$-\frac{2}{3}$
$\frac{2}{3}$	0	0	$\frac{3}{2}$	1	$-\frac{1}{6}$	$-\frac{1}{3}$
$\frac{2}{3}$	1	0	0	0	$\frac{1}{3}$	$-\frac{1}{3}$
$\frac{2}{3}$	0	1	$\frac{1}{2}$	0	$-\frac{1}{6}$	$\frac{2}{3}$

Vi har dermed, at det duale problem har løsningen

$$y_1 = 0,\ y_2 = \frac{1}{3},\ y_3 = \frac{2}{3}.$$

Bemærk, at første variabel har værdien 0, svarende til, at den indgik i den optimale løsning til det oprindelige problem. Det svarer fint til intutionen omkring skyggepriser som værdien af, at bibetingelsen slækkes en smule: Hvis bibetingelsen ikke binder i den optimale løsning, har man heller ikke nogen glæde af, at der bliver lidt mere "luft" i denne betingelse.

7. Opgaver

1. Løs det lineære programmeringsproblem

$$\begin{aligned}&\max 2x_1 + 3x_2 + 3x_3 + x_4\\ &\text{under bibetingelserne}\\ &x_1 + x_2 + x_3 = 10\\ &x_1 + 2x_3 = 8\\ &x_2 + x_4 = 7\\ &x_1, x_2, x_3, x_4 \geq 0\end{aligned}$$

ved simplexmetoden. Bemærk: Der er ikke nogen basis i slackvariable. Brug standardmetoden til at finde en.

Den næste opgave er et eksempel på såkaldt *målprogrammering*, hvor man tænker sig, at beslutningstageren har flere mål, som han/hun ønsker opfyldt, men som ikke kan opfyldes samtidig. Dette kan ofte med fordel formuleres som et LP-problem, hvor hvert af målene giver en bibetingelse, der så kan over- eller underopfyldes. Det sidste opnås ved at tilføje to slackvariable i hver bibetingelse. Beslutningstagerens holdning til over- eller underopfyldelse søges da indfortolket i kriteriefunktionen.

2. En virksomhed producerer tre varer i årlige mængder x_1, x_2, x_3. Disse varer har dækningsbidrag på henholdsvis 12, 9 og 15 pr. enhed, og der er beskæftiget henholdsvis 5,2 og 4 medarbejdere pr. enhed. Endelig er der et kapitalkrav (i form af nyinvestering) på 5, 7 og 8 pr.enhed.

Virksomheden har en målsætning gående ud på, at profitten ihvertfald skal være på 125, at beskæftigelsen skalæ være 40, og at investeringerne ikke må overstige 50.

Undersøg om målsætningerne kan opfyldes samtidig. Hvis ikke, har ledelsen meddelt, har ledelsen givet pointværdi pr.enhed afvigelse (idet større pointtal er udtryk for dårligere situation): Underopfyldelse af profitmål: 5, Overopfyldelse af beskæftigelsesmål: 2, Underopfyldelse af samme: 3, Overopfyldelse af investeringsmål: 3. De ikke anførte muligheder vurderer ledelsen som uden ulemper.

Formuler som LP og find løsning.

3. En klinik behandler tre typer patienter ved genoptræning på tre forskellige maskiner; en standardbehandling på en patient af hver type lægger beslag på maskinerne i overensstemmelse med tabellen nedenfor:

Patienttype:	1	2	3
Maskine:			
I	4	7	9
II	5	5	4
III	8	7	12

hvor tallene er opgivet i minutter. Klinikken er åben i 7 timer om dagen.

Klinikken drives af en velgørende institution, som selv visiterer patienter til behandling. Målsætningen er at gøre antallet af patienter i den gruppe, der behandles færrest af, så stort som muligt.

Opstil problemet som et LP-problem og løs det.

4. Løs LP-problemet

$$\max 2x_1 + 3x_2 + 5x_3$$

under bibetingelserne

$$3x_1 + 2x_2 + x_3 \leq 4$$
$$x_1 + x_2 + x_3 = 8$$
$$x_1 + 7x_2 \geq 16$$
$$x_1, x_2, x_3 \geq 0$$

5*. Et *transportproblem* (som vi skal se meget mere på senere) går ud på at minimere samlede transportomkostninger ved at flytte givne varemængder $s_1, \ldots, s_m$ i m *kilder* til fastlagte efterspøgsler $t_1, \ldots, t_m$ i n *terminaler*, idet $\sum_{i=1}^{m} s_i = \sum_{j=1}^{n} t_j$. Omkostningerne ved at transportere en enhed fra kilde i til terminal j er c_{ij}.

Formuler transportproblemet som et LP problem og opstil starttableauet.

Formuler det duale problem. Vis, at hvis man til en basisløsning (x_{ij}^0) i det oprindelige problem kan finde tal $\mu_1, \ldots, \mu_m$ og $\nu_1, \ldots, \nu_n$ så

$$\mu_i + \nu_j \leq c_{ij}$$

med lighedstegn hvis x_{ij}^0 er forskellig fra 0, da er løsningen (x_{ij}^0) optimal.

Her er lidt simplex-træning:

6. En virksomhed fremstiller fire produkter, I, II, III og IV. Om disse produkter gælder følgende:

	I	II	III	IV
Salgspris, kr.pr.stk.	15	72	105	30
Stykomkostn., kr.pr.stk.	5	17	30	20
Maskintid,min.pr.stk.	1	2	3	0
Elforbrug,kWh pr.stk.	2	40	20	10

Virksomheden råder over en maskinpark, der kan arbejde i 3.000 timer pr.måned. Af historiske årsager er virksomheden underkastet en rationering i elforbruget, der ikke må overstige 50.000 kWh om måneden. Endelig er der et affaldsprodukt, hvoraf der opstår 1 mg pr. produceret enhed af vare I og II, men ikke af III og IV. Der må maximalt udledes 500 mg af dette stof om måneden.

(1) Find optimal produktion i virksomheden.

(2) Hvis der udbydes tilladelser til yderligere udledning, hvor meget vil virksomheden være villig til at ofre pr. mg?

8. Litteratur

Lineær programmering er særdeles grundigt beskrevet i litteraturen. Alternative fremstillinger er f.eks. Hillier og Lieberman (1990). En ældre, men stadig fremragende teoretisk gennemgang af dette (og tilstødende) emner findes i Gale (1960).

KAPITEL 3

Mere om lineær programmering

1. Indledning

I det foregående kapitel har vi koncentreret os om simplex-algoritmen. I det følgende ser vi på en række tilføjelser og udvidelser. I første omgang er der især tale om alternative måder at gøre det samme på (dual simplex og revideret simplex), men vi skal også se nærmere på et par helt alternative forslag til, hvordan man løser lineære programmeringsproblemer (ellipsoid-metoden og Karmarkers metode).

Lineær programmering er arbejdshesten i kvantitative metoder indenfor økonomi, og der er derfor ingen mangel på anvendelser. I afsnit 5 ser vi nærmere på en anvendelse, som har at gøre med et emne, der har aldrig svigtende aktualitet, nemlig produktivitet i den offentlige sektor. Det drejer sig om den såkaldte DEA-analyse, der er en metode til rekonstruktion af teknologien i en virksomhed udfra et givet datasæt, knyttet sammen med en måling af, hvorvidt en given produktion er efficient. Det er i rekonstruktionsdelen, at der bruges lineær programmering, og den anvendte metode har muligheder også udenfor den konkrete anvendelse.

2. Dual simplex-metode

Dualitetsbetragtninger kan bruges til andet end at aflæse skyggepriser; der er situationer, hvor man med fordel kan vende hele problemet og løse det ved simplex set fra den duale synsvinkel.

Grunden til, at der opstår et behov for at løse LP-problemet på en anden måde, er, at betingelserne for simplex (i sædvanlig forstand) kan være overtrådt. Nærmere bestemt kan der optræde situationer, hvor vi har en løsning, som godt nok er optimal (alle c-koefficienter negative), men som ikke er brugbar: Som tidligere nævnt kræves det hertil, at de koefficienter, der står i b-søjlen, alle er ikke-negative. Det kan måske umiddelbart være uklart, hvordan man kan have rodet sig ind i sådan en situation, men den opstår ret naturligt i visse forbindelser; det skal vi senere se eksempler på.

Som regel kan man klare sådanne problemer ved at lave passende mange rækkeoperationer; det er imidlertid temmelig kluntet og ikke egnet til systematiske

løsningsmetoder. Her kommer dual simplex ind.

Fremgangsmåden er egentlig ganske simpel: Vi skal ligesom i sædvanlig simplex skifte basissøjler ud, men vi vender proceduren om: I stedet for at hente den "bedste" variabel udenfor basis ind, smider vi den "værste" variabel i basis ud. Det, vi så skal finde ud af, er at erstatte med en passende ny variabel, således at løsningen forbliver optimal, men at negativiteten af koefficienter er forbedret. Man gør følgende:

1. Brugbarhedsbetingelse: Fjern den variabel i løsningen, som har den største negative koefficient.

2. Optimalitetsbetingelse: Vælg *rækken* hørende til variablen valgt ovenfor, og betragt for hver *søjle,* hvor der står noget *negativt*, det tilsvarende element i c-rækken divideret med elementet i søjlen. Da begge tal er negative, er resultatet > 0. Vælg variabel svarende til det *mindste* af disse tal.

Efter at have fundet pivotelement (svarende til en række, der går ud, og en søjle, der går ind) gennemføres helt sædvanlige rækkeoperationer, hvorved tableauet opdateres.

Eksempel 3.1

Betragt LP-problemet

$$\min 2x_1 + x_2$$

under bibetingelserne

$$3x_1 + x_2 \geq 3$$
$$4x_1 + 3x_2 \geq 6$$
$$x_1 + 2x_2 \leq 3$$
$$x_1,\ x_2 \geq 0$$

Vi opstiller simplex-tableauet, idet der skiftes fortegn på c-koefficienterne (jvf. afsnit 4) og der indføjes slackvariable i de tre bibetingelser (med fortegnsskift svarende til $\geq$). Vi får simplex-tableuet

	-2	-1	0	0	0
-3	-3	-1	1	0	0
-6	-4	-3	0	1	0
3	1	2	0	0	1

De negative elementer i b-søjlen viser, at vi ikke har en mulig løsning. På den anden side viser c-rækken, at optimalitetsbetingelsen er i orden. Vi kan derfor bruge dual simplex.

Eksempel 3.1, fortsat

Vi ønsker at fjerne 2. række, svarende til variablen x_4. For at finde ud af, hvilken variabel, som skal ind i stedet, sammenligner vi tallene

$$\frac{-2}{-4}, \frac{-1}{-3};$$

alle øvrige brøker falder bort på grund af fortegnsbetingelserne. Da den sidste af dem er mindst, skal vi vælge denne. Pivotelementet er 2.række, 2. søjle, og det nye tableau bliver:

2	$-\frac{2}{3}$	0	0	$-\frac{1}{3}$	0
-1	$-\frac{5}{3}$	0	1	$-\frac{1}{3}$	0
2	$\frac{4}{3}$	1	0	$-\frac{1}{3}$	0
-1	$-\frac{5}{3}$	0	0	$\frac{2}{3}$	1

Vi har stadig ikke en brugbar løsning, så vi fortsætter. Lader vi x_3 (øverste række) udgå får vi, at x_1 skal ind, og det nye tableau bliver:

$\frac{12}{5}$	0	0	$-\frac{2}{5}$	$-\frac{1}{5}$	0
$\frac{3}{5}$	1	0	$-\frac{3}{5}$	$\frac{1}{5}$	0
$\frac{6}{5}$	0	1	$\frac{4}{5}$	$-\frac{3}{5}$	0
0	0	0	-1	1	1

Løsningen er nu både optimal og mulig.

Der er egentlig ikke noget overraskende i, at vi kan løse LP-problemer på denne måde. Faktisk er den duale simplex jo blot simplex-metoden anvendt på det duale problem. Det nye er faktisk kun, at man under visse omstændigheder med fordel kan starte med almindelig simplex og senere skifte over til dual simplex. Det skal vi se et eksempel på i et senere kapitel.

3. Revideret simplex

Som vi har set i det forrige kapitel, arbejder simplex-algoritmen sig gennem successive omformninger eller opdateringer af et tableau, hvis størrelse er $(m + 1) \times (n + 1)$, hvor m og n er antallet af henholdsvis rækker og søjler i matricen af bibetingelser, når problemet er bragt på standard form. Det vil sige, at m er antallet

af bibetingelser i det oprindelige problem, mens n er antal oprindelige variable plus de eventuelt yderligere tilføjede slack-variable.

Det er ret oplagt, at et given udgave af LP-problemet vil være mere besvælig at løse, jo større m og n bliver (en mere præcis formulering af dette kommer senere, når vi diskuterer *beregningskompleksitet)*. Det fører på sin side til, at vi interesserer os for simplere omformuleringer af proceduren i simplex, noget som af og til vil gøre det muligt at håndtere "store" problemer på en forholdsvis simpel måde. Vi skal senere se eksemler på, at LP-problemer med en særlig struktur kan løses på anden og nemmere måde end med simplex. Her vil vi se på en generel metode til at systematisere simplex-beregningerne. Det er stadigvæk simplex, som gennemføres, men med et væsentlig mindre tableau. Man sparer opdateringen af en masse variable, der viser sig ikke at skulle bruges til noget. Ved en omhyggelig gennemgang af princippet i simplex-algoritmen vil det ses, at man faktisk ikke behøver at opdatere hele tableauet i hvert beregningstrin. Man kan nøjes med at opdatere i et tableau med $m+1$ rækker og $m+1$ søjler, idet de resterende $n-m$ søjler kan rekonstrueres i det omfang, man har brug for dem. Og pointen er, at man *kun har brug for én ad gangen,* nemlig den, der skal ind i basis.

Vi skriver (som sædvanlig) vort LP-problem som

$$
\begin{aligned}
&\max c_1x_1 + \cdots + c_nx_n \\
&\text{under bibetingelserne} \\
&a_{11}x_1 + \cdots + a_{1n}x_n = b_1 \\
&\qquad\vdots \\
&a_{m1}x_1 + \cdots + a_{mn}x_n = b_m \\
&x_1, \ldots, x_m \geq 0
\end{aligned}
$$

og antag, at koefficientmatricen er sådan, at de sidste m søjler udgør en enhedsmatrix (sådan som det vil være, hvis bibetingelserne oprindelig var givne ved ulighedstegn, og vi derefter tilføjede slack-variable).

Vort første tableau i den reviderede simplex-metode består da af følgende:

$-z_0$	0	$\cdots$	0
$\hat{b}_1$	1	$\cdots$	0
$\vdots$	$\vdots$		$\vdots$
$\hat{b}_m$	0	$\cdots$	1

hvor vi kun har taget søjler svarende til slackvariablene (eller generelt: svarende til vor første basis) med, samt randsøjlen med b_i'erne.

Vi noterer os nu, at efter et vilkårligt antal basisskift i den oprindelige simplexmetode vil vi, hvis B er delmatricen af A svarende til den basis $\mathcal{B}$, som vi er nået frem til, på pladserne i tableauet svarende til søjlerne $x_{n-m+1}, \ldots, x_n$, dvs. svarende til første basis, have matrcen B^{-1}.

For at få koefficienterne i øverste, 0'te, række hørende til søjle j skal vi som bekendt udregne

$$\bar{c}_j = c_j - z_j = c_j - \sum_{i \in \mathcal{B}} x_{ij} c_{B(i)}.$$

For søjlerne svarende til $x_{n-m+1}, \ldots, x_n$ får vi, at vektoren z af z_j'er kan findes ved

$$z = c_B B^{-1},$$

hvor $c_B = (c_{B(1)}, \ldots, c_{B(m)})$, og for de øvrige søjlæer fås

$$z_j = c_B B^{-1} A_j,$$

hvor A_j er den j'te søjle i matricen A. Det ses heraf, at alle koefficienter i øverste række kan rekonstrueres udfra kendskab til det allerførste tableau samt matricen B^{-1}, dvs. den del af det opdaterede tableau, der svarer til de sidste m søjler. Klart nok kan vi også rekonstruere de manglende søjler (det har allerede været brugt implicit ovenfor), nemlig ved

$$x_j = B^{-1} A_j.$$

Vi kan nu skitsere gangen i *revideret simplex:* Dert startes med det oprindelige tableau og den del, der skal benyttes fremover (vist ovenfor), og som vi her vil kalde $G_{(0)}$. Det første basisskift findes som vanligt i det komplette tableau.

I hvert trin h sker der en opdatering som følger: Udgangspunktet er $(m+1) \times (m+1)$-tableauet $G_{(h)}$ på formen

$-z_0$	$-\pi$
b	B^{-1}

Herfra beregnes følgende:

(1) De manglende c-koefficienter findes af

$$\bar{c}_j = c_j - \pi A_j, \ \pi = c_B B^{-1},$$

og herfra findes som sædvanlig en søjle med positiv $\bar{c}_j$, som skal ind i basis – eller vi har en optimal løsning.

(2) De manglende koefficienter i den søjle s, som skal ind i basis, findes fra

$$x_s = B^{-1} A_s.$$

(3) Pivot-elementet x_{rs} i søjlen X_s findes som sædvanlig fra betingelsen

$$\frac{\bar{b}_r}{x_{rs}} = \min_{i:x_{is}>0} \left\{ \frac{\bar{b}_i}{x_{is}} \right\}.$$

(4) Tableauet $G_{(h-1)}$ opdateres nu til $G_{(h)}$ på samme måde som ved almindelig opdatering.

fordelen ved metoden – der som nævnt i starten ikke så meget er en metode i sig selv som en særlig måde at gennemføre simplex-algoritmen på – er som allerede nævnt at man i hvert trin kan nøjes med at operere med et $(m+1) \times (m+1)$ tableau. Dette har især betydning i problemer, hvor n er meget stor i forhold til m. Videre gælder, at mange problemer, der fra naturens hånd ikke ser ud til at have et n særlig meget større end m, ved en passend eomformulering kan gøres til et problem med mindre m, omend muligvis med langt større n. Et eksempel på dette er den såkaldte Dantzig-Wolfe dekomponering, som vil blive beskrevet i næste afsnit.

4. Dekomponering af LP-problemer

Som vi har været inde på ved flere lejligheder, er simplex-algoritmen en løsningsmetode, der set i sin helhed (dvs. på baggrund af en bred vifte af de optimeringsopgaver, som man kommer ud for i praksis) giver det ønskede resultat hurtigst muligt. Men der kan selvfølgelig være situationer, hvor problemet er så omfattende, at det i praksis er vanskeligt for ikke at sige umuligt at gennemføre alle beregningstrin.

I sådanne situationer vil man typisk søge efter særlige optimeringsmetoder, der på forskellig måde skyder genvej i beregningerne. Hertil må man selvfølgelig udnytte særlige egenskaber ved deforeliggende problem, og det er kun, hvis problemet har en særlig struktur, at man kan korte det ned til et mindre og mere håndterligt.

I dette afsnit vi se på et sådant tilfælde, som endda kan gives en særlig fortolkning; optimeringen kan anskues som en hierarkisk procedure, hvor et center giver passende budskaber til sine underordnede afdelinger, der benytter dette budskab til at optimere et afdelingsspecifikt kriterie. Resultatet af denne optimering meddeles til centret, der checker det mod sit overordnede kriterie ogmodificerer budskaberne herefter.

Proceduren betegnes som *dekomponering* af det oprindelige optimeringsproblem. De underordnede afdelingers kriterier vil afhænge af det overordnede kriterium på en bestemt, nøje specificeret måde, så fortolkningen skal altså ikke presses for langt: Der er ikke tale om en *decentralisering* af beslutninger, idet man

så måtte tage hensyn til de faktiske målsætninger i afdelingerne og sørge for, at de procedurer, der indføres, er *incitamentforenelige,* noget som er en væsentlig mere kompliceret sag. Hvad vi gør her, er stadig en rent mekanisk procedure, som blot udnytter, at det er nemmere at løse en række små problemer end et enkelt meget stort.

Antag at vort LP problem har formen

$$\begin{aligned}
&\max z = c \cdot x + d \cdot y \\
&\text{under bibetingelserne} \\
&Dx + Fy = b_0 \\
&Ax \qquad = b_1 \\
&\qquad By = b_2 \\
&x, y \geq 0.
\end{aligned}$$

Her er x en n_1-vektor, y en n_2-vektor, og D, F, A og B har de relevante dimensioner (hvilket vil sige, at D er en $(m_0 \times n_1)$-matrix, hvor m_0 er antallet af fælles bibetingelser, F en $(m_0 \times n_2)$-matrix; videre har A dimensionen $(m_1 \times n_1)$, og B er en $(m_2 \times n_2)$-matrix, hvor m_1 og m_2 udgør antallet af specifikke bibetingelser for hver af de to afdelinger. Det betyder, at den samlede bibetingelsesmatrix har en blokstruktur af formen

$$\begin{pmatrix} D & F \\ A & 0 \\ 0 & B \end{pmatrix},$$

således at der kommer 0-matricer ind på pladserne svarende til de specifikke bibetingelser.

Lad os et øjeblik overveje, hvad vi skulle gøre i en særlig nem situation, nemlig når der slet ikke er fælles bibetingelser. I det tilfælde vil det samlede maximum, fremkommet ved valg af x og y så at det lineære kriterie får størst mulig værdi, kunne findes ved at løse to små LP-problemer hver for sig, idet valget af x ikke på nogen måde indskrænker valget af y.

Helt så simpelt slipper vi ikke i det generelle tilfælde, hvor der netop er m_0 bibetingelser, som er fælles for de to delproblemer; man kan derfor ikke løse dem enkeltvis og regne med, at resultatet også er optimalt i den større helhed. Men det kan på dan anden side tænkes, at det forholdsvis simpelt vil kunne lade sig gøre at tilpasse de fælles restriktioner i løbet af nogle iterationer; det er netop fremgangsmåden i den følgende algoritme, der kaldes for *Dantzig-Wolfe-dekomponering*.

Vi betragter to delproblemer, hvor matricen af bibetingelser er henholdvis A og B. Delproblem I har bibetingelserne

$$Ax = b,\ x \geq 0.$$

Geometrisk er dette, som vi har været inde på, en konveks polytop udspændt af passende punkter $x^1, \ldots, x^p$ i $\mathbb{R}^{n_1}$, hvilket omvendt betyder, at enhver løsning til

bibetingelserne kan skrives som en konveks kombination af disse punkter,

$$x = \sum_{i=1}^{p} \lambda_i x^i,\ \lambda_i \geq 0,\ i = 1, \ldots, p,\ \sum_{i=1}^{p} \lambda_i = 1. \tag{1}$$

Tilsvarende er de mulige løsninger til delproblem II udspændt af punkter $y^1, \ldots, y^q$ i $\mathbb{R}^{n_2}$ og løsning y til disse bibetingelser kan skrives

$$y = \sum_{j=1}^{q} \mu_j y^j,\ \mu_j \geq 0,\ j = 1, \ldots, q,\ \sum_{j=1}^{q} \mu_j = 1. \tag{2}$$

Med denne formulering af bibetingelserne kan vi omskrive det oprindelige LP-problem, idet vis sætter $\xi_i = c \cdot x^i,\ i = 1, \ldots, p$, $\delta_j = d \cdot y^j$, $j = 1, \ldots, q$, og indsætter (1) og (2):

$$\begin{aligned} &\max \xi \cdot \lambda + \delta \cdot \mu \\ &\text{under bibetingelserne} \\ &\Delta\lambda + \Phi\mu = b_0 \\ &e \cdot \lambda \qquad = 1 \\ &\qquad e \cdot \mu = 1 \\ &\lambda, \mu \geq 0, \end{aligned} \tag{3}$$

hvor vi har brugt notationen Δ for matricen med søjler $\Delta_j = Dx^j$, $j = 1, \ldots, p$, Φ for matricen med søjler $\Phi_j = Fy^j$, $j = 1, \ldots, q$, og hvor e betegner en vektor af ettaller (der i formuleringen ovenfor optræder to gange med forskellig dimension, nemlig henholdsvis p og q). I denne udgave kaldes problemet normalt for *master-problemet*.

I forhold til den oprindelige formulering af optimeringsproblemet er der i første omgang ihvertfald sket det, at der er kommet nye variable, nemlig λ og μ i stedet for x og y. Da dimensionen af de nye variable svarer til antallet af hjørner i de konvekse polytoper – og sådanne hjørner vil der som regel være temmelig mange af – der udgøres af løsningerne, virker det egentlig nærmest som om vi har gjort problemet mere kompliceret, end det var fra starten. Det er nu ikke helt rigtigt, for til gengæld for de mange variable har vi fået færre bibetingelser; der er nemlig nu kun én afdelingsspecifik bibetingelse for hver afdeling. Denne reduktion i antal bibetingelser er faktisk vigtigere end forøgelsen i antallet af variable: Vi har netop set, at med revideret simplex behøver man kun at holde styr på et tableau af størrelsen $(m+1) \times (m+1)$, hvor m er antal bibetingelser; antallet af variable er ikke helt ligegyldigt (man skal jo trods alt søge gennem samtlige c-koefficienter ved hvert basisskift), men det har slet ikke samme vægt i kompleksiteten som antallet af bibetingelser. Omskrivningen er derfor et vigtigt skridt fremad mod en reduktion af problemet.

Der gås frem på følgende måde: Antag, at vi er kommet til h'te trin i revideret simplex på master-problemet, hvor vi har et tableau $G_{(h)}$ på følgende form:

$-z_0$	$-\pi$	$-\alpha$	$-\beta$
b			

hvor tableauets inddeling antyder de m fælles og de to afdelingsspecifikke bibetingelser.

Når vi skal finde en ny søjle til basis, skal vi ifølge trin (1) i revideret simplex først udregne alle de opdaterede c-koefficienter. For søjlerne svarende til delproblem I er det

$$\begin{aligned}\hat{\xi}_j &= \xi_j - \begin{pmatrix}\pi & \alpha & \beta\end{pmatrix}\begin{pmatrix}\Delta_j \\ 1 \\ 0\end{pmatrix} \\ &= \xi_j - \pi\Delta_j - \alpha,\end{aligned}$$

$j = 1, \ldots, p$. At vælge en søjle, hvor $\xi_j > 0$ svarer dermed til at vælge j således at $\xi_j - \pi\Delta_j > \alpha$.

Vi behøver imidlertid ikke at finde $\xi_j - \pi\Delta_j$ for hvert j; vi skal jo alligevel højst bruge en enkelt af dem, nemlig den største, så det er naturligt straks at lede efter denne, hvilket vil sige at løse problemet

$$\max \xi_j - \pi\Delta_j = \max (c - \pi D) \cdot x^j,$$

hvor der søges over alle hjørner x^j i den konvekse polytop af mulige løsninger til delproblem I. Men dette blot en lidt teknisk måde at formulere et LP-problem på, nemlig problemet

$$\begin{aligned}&\max (c - \pi D) \cdot x \\ &\text{under bibetingelsen} \\ &Ax = b_1, \\ &x \geq 0.\end{aligned} \tag{4}$$

Helt tilsvarende kan vi for søjler svarende til delproblem II finde den mest lovende søjle ved at løse

$$\begin{aligned}&\max (d - \pi F) \cdot y \\ &\text{under bibetingelsen} \\ &By = b_2, \\ &y \geq 0.\end{aligned} \tag{5}$$

Hvis (4) har en løsning x^j med kriterieværdi større end α kan vi regne os frem til den pågældende søjles koordinater i master-problemet og fortsætte revideret simplex. Har (4) ikke nogen løsning, prøver vi i stedet med (5). Hvis heller ikke denne har nogen løsning, kan vi slutte, at alle c-koefficienter i master-problemet er ikke-positive, og vi har dermed en optimal løsning til det oprindelige problem.

Fortolkning af dekomponeringsalgoritmen: Dantzig-Wolfe-metoden involverer to typer af LP-problemer, nemlig dels master-problemet, i hvilket der for hver iteration findes c-koefficienter for visse søjler, dels de enkelte delproblemer, hvor der findes en optimal løsning til et LP-problem, hvis kriteriefunktion bestemmes gennem vektoren (π, α, β).

Som sædvanlig har de duale variable en fortolkning i form af priser. De relative gevinstkoefficienter i m_0-vektoren π kan opfattes som (skygge-)priser på de fælles bibetingelser for afdelingerne. Centret overlader det til hver af afdelingerne at maximere gevinst, idet det kræves, at afdelingerne i deres afvejning af gevinsten ved alternative aktiviteter skal inddrage hensynet til de fælles bibetingelser, der kommer til udtryk i et fradrag af størrelsen πD ("for træk på fælles ressourcer") for afdeling I og på πF for afdeling II. Vi kan altså opfatte de enkelte koefficienter i π som *interne afregningspriser* hørende til hver af bibetingelserne.

På tilsvarende måde kan koefficienterne α og β gives en fortolkning, nemlig som den gevinst, der kan skabes i henholdsvis afdeling I og II ved at køre de aktiviteter, som centret aktuelt har kalkuleret med, og i det aktuelt besluttede omfang. Denne meddelelse om planlagt gevinst sendes ud til afdelingerne sammen med de interne afregningspriser, og det er nu op til afdelingerne at undersøge, om de kan gøre det bedre. Hvis de kan det, sendes et konkret forslag tilbage til centret. I næste trin undersøges det, hvilke konsekvenser et sådant forslag fra en afdeling får for de interne afregningspriser og gevinstkravene for hver afdeling. Kommunikationen mellem center og afdelinger kræver kun transmission af vektoren (π, α, β) (der kan opfattes som "prisinformation" den ene vej af en plan x^j eller y^j (mængdeinformation) den anden vej. Med denne information kan parterne selv regne sig frem til resten.

5. En anvendelse af lineær programmering: DEA-analyse

I dette afsnit vil vi se lidt nærmere på en anvendelse af lineær programmering ved analysen af *produktiviteten* i en virksomhed. Begrebet produktivitet er blandt den økonomiske teoris mest centrale, og det bruges også meget aktivt i mere praktisk orienterede sammenhænge. Hvad der skal forstås ved produktivitet udenfor de ret snævre rammer, der sættes af teoriens brug af begrebet (som regel i forbindelse med en given veldefineret produktionsfunktion), er dog som regel noget uklart. Der er en lang tradition for at måle produktivitet som et passende forhold mellem output og input, men denne tradition er det ofte svært at leve op til, for såvel input som

output vil i mange tilfælde vlre et sortiment bestående af mange varer, herunder endda sådanne, som der ikke kan sættes nogen oplagt pris på, fordi det er varer, der ikke sælges på et marked. Mange undersøgelser af produktiviteten i den offentlige sektor rammer ind i dette problem, idet output ikke sælges til publikum men snarere består af ydelser, der stilles gratis til rådighed for brugerne.

Da det, som man ved, alligevel er vigtigt at kunne vurdere produktiviteten (i den løse forstand: sammenhængen mellem det producerede varesortiment og den hertil brugte mængde af inputs af forskellig art), må man finde på noget nyt. En af de mest brugte metoder er den såkaldte DEA-analyse (DEA står for Data Envelopment Analysis, så "DEA-analyse" er ikke nogen helt smart betegnelse, men den har nu en gang vundet indpas), introduceret af Charnes, Cooper og Rhodes i 1978.

Vi antager, at den virksomhed, som vi skal undersøge, producerer et sortiment $(y_1, \ldots, y_s)$ bestående af s forskellige varer, hvortil den benytter m forskellige slags input, $x_1, \ldots, x_m$. Da der er flere varer, må vi foretage en sammenvejning såvel på input- som outputsiden. Betegnes de vægte, der benyttes, med $u_1, \ldots, u_s$ på outputsiden og $\nu_1, \ldots, \nu_m$ for inputs, kan vi operere med et forhold mellem sammenvejet output og input som udtryk for produktiviteten,

$$h = \frac{\sum_{r=1}^{s} u_r y_r}{\sum_{i=1}^{m} \nu_i x_i}.$$

Vi er imidlertid foreløbig ikke kommet rigtig ud af starthullerne, for vor brøk her er ikke bedre end de vægte, som blev brugt ved sammenvejningen, og de er indtil videre helt vilkårlige.

Men lad os nu antage, at vi ikke skal vurdere virksomhedens produktivitet helt løsrevet fra omverdenen, men at vi har et *sammenligningsgrundlag,* som kan være andre helæt tilsvarende virksomheder (hvis der er sådanne) eller virksomheden selv i de foregående perioder (idet det så må forudsættes, at de teknologiske betingelser for produktionen ikke har ændret sig). Helt præcist vil vi antage, at der er n andre observationer af såvel input som output,

$$(x_j; y_j) = (x_{j1}, \ldots, x_{jm}; y_{j1}, \ldots, y_{js}), \ , j = 1, \ldots, n,$$

således at vor opgave er at vurdere den betragtede virksomhed, som vi nu vil betegne med index 0, på baggrund af de øvrige n virksomheder.

På denne baggrund virker rimeligt at vælge vægte, så at forholdet h er mindre end 1 for samtlige n referencevirksomheder (det giver en slags normering af produktivitetsmålet). Denne betingelse vil dog ikke alene bestemme de vægte, der skal bruges; men så kan man f.eks. vælge blandt de mulige på en sådan måde, at resultatet bliver så favorabelt som muligt for den konkret betragtede. Det fører til, at det søgte udtryk for produktiviteten i virksomhed 0 findes som løsning til

maximeringsproblemet

$$\begin{aligned}
&\max h_0 = \frac{\Sigma_{r=1}^{s} u_r y_{0r}}{\Sigma_{i=1}^{m} \nu_i x_{0i}} \\
&\text{under bibetingelserne} \\
&\frac{\Sigma_{r=1}^{s} u_r y_{jr}}{\Sigma_{i=1}^{m} \nu_i x_{ji}} \leq 1,\ j = 1, \ldots, n, \\
&u, \nu \geq 0.
\end{aligned} \tag{1}$$

Vi har nu reduceret problemet om at måle produktivitet til et optimeringsproblem. Uheldigvis er det *ikke* et LP-problem, for både det kriterie, der skal optimeres, og udtrykket på venstre side i bibetingelserne, er ikke-lineært i de variable, u og ν, som der skal maximeres over. Der er dog ingen grund til at fortvivle: Det hænder ikke sjældent, at problemer, som ikke umiddelbart ser ud som LP-problemer, efter en behændig omformulering kommer på den rette form. Således også her; vi starter med at betragte den helt ækvivalente formulering af problemet (1), hvor vi blot har vendt alle brøker; den får formen

$$\begin{aligned}
&\min f_0 = \frac{\Sigma_{i=1}^{m} \nu_i x_{0i}}{\Sigma_{r=1}^{s} u_r y_{0r}} \\
&\text{under bibetingelserne} \\
&\frac{\Sigma_{i=1}^{m} \nu_i x_{ji}}{\Sigma_{r=1}^{s} u_r y_{jr}} \geq 1,\ j = 1, \ldots, n, \\
&u, \nu \geq 0.
\end{aligned} \tag{2}$$

Herfra kommer vi videre ved at se lidt nærmere på bibetingelserne. Hvis der for hvert j skal gælde, at forholdet mellem $\Sigma_{i=1}^{m} \nu_i x_{ji}$ og $\Sigma_{r=1}^{s} u_r y_{jr}$ skal være større end eller lig 1, så må *forskellen* mellem dem være ikke-negativ (og omvendt), så vi kan omskrive (2) til

$$\begin{aligned}
&\min f_0 = \frac{\Sigma_{i=1}^{m} \nu_i x_{0i}}{\Sigma_{r=1}^{s} u_r y_{0r}} \\
&\text{under bibetingelserne} \\
&\sum_{i=1}^{m} \nu_i x_{ji} - \sum_{r=1}^{s} u_r y_{jr} \geq 0,\ j = 1, \ldots, n, \\
&u, v \geq 0,
\end{aligned} \tag{3}$$

hvorved vi ihvertfald kom af med ikke-lineariteten i bibetingelserne.

For at slippe af med brøken i kriteriefunktionen indfører vi en normering: Man kan se, at hvis vi ganger såvel u'er som v'er med samme konstant K, så ændres

ingenting i værdien af f_0. Derfor kan vi nøjes med at søge over (u, v) normeret på passende måde, og vi vælger at holde os til alle de (u, v), som opfylder

$$\sum_{r=1}^{s} v_r y_{0r} = 1.$$

Derved kommer der til at stå 1 i nævneren af brøken f_0 uanset hvilke (u, v) vi vælger (så længe normeringen holder), og så kan vi jo lige så godt glemme, at der var tale om en brøk! Ialt har vi derfor omformuleringen af det oprindelige problem til problemet

$$\begin{aligned}
&\text{min } \Sigma_{i=1}^{m} \nu_i x_{0i} \\
&\text{under bibetingelserne} \\
&\sum_{i=1}^{m} \nu_i x_{ji} - \sum_{r=1}^{s} u_r y_{jr} \geq 0,\ j = 1, \ldots, n, \\
&\sum_{r=1}^{s} v_r y_{0r} = 1, \\
&u, v \geq 0.
\end{aligned} \tag{4}$$

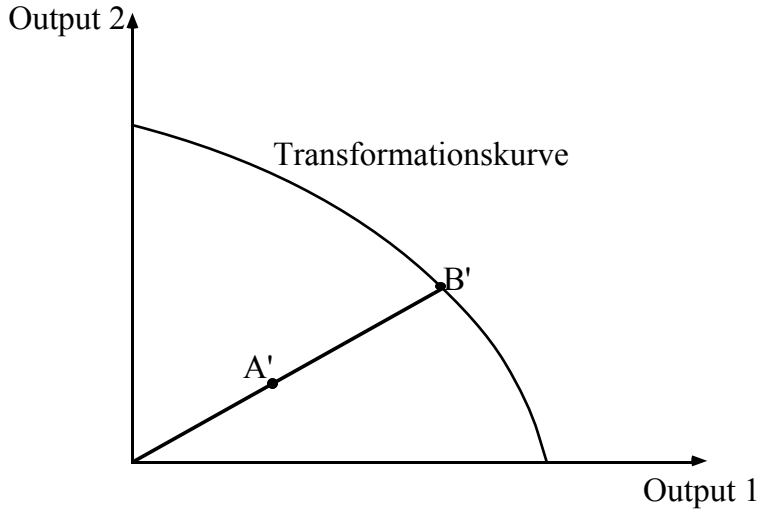
Vi har nu et helt normalt LP-problem, som vi kan løse; kriterieværdien (eller, for at være helt præcis, den reciprokke af kriterieværdien) giver det ønskede udtryk for produktiviteten målt relativt til sammenligningsgrundlaget.

Der kan være behov for at se DEA-metoden i et lidt andet – måske lidt mere teoretisk orienteret – lys for at overbevise sig om, at den konkret valgte metode, gående ud på at tilpasse vægte så at forholdet mellem sammenvejet output og input bliver maximalt, givet at det var ≤ 1 i referencevirksomhederne, ikke er grebet ud af luften. Til det formål kan det være praktisk at se vor fremgangsmåde som en procedure, der egentlig består af to helt adskilte dele, nemlig

(1) indsamling af kendskab til den konkrete virksomheds produktionsmulighedsområde,

(2) måling af efficiens af den konkrete gennemførte produktion i forhold til de tekniske muligheder, som repræsenteret ved produktionsmulighedsområdet.

Lad os i den følgende diskussion simplificere ved at antage, at der kun bruges én inputvare (det lyder umiddelbart som en meget uheldig antagelse, men det er ikke helt så slemt, for det, det egentlig kommer an på, er om der findes en oplagt måde at aggregere de forskellige varer på, og det er ikke så sjældent, at inputsiden kan værdisættes til observerede markedspriser), og at der er konstant skalaafkast i produktionen. Vi kan da normere alle de observerede produktioner $(x_j; y_{j1}, \ldots, y_{js})$, $j = 0, 1, \ldots, n$, så at $x_j = 1$ (og vi betrager derme output pr. krones input i produktionen). Geometrisk kan de mulige outputkombinationer afbildes som en mængde af punkter liggende indenfor en transformationskurve, således som vist i figur 1. Problemet om at måle en given gennemført produktions



Figur 1

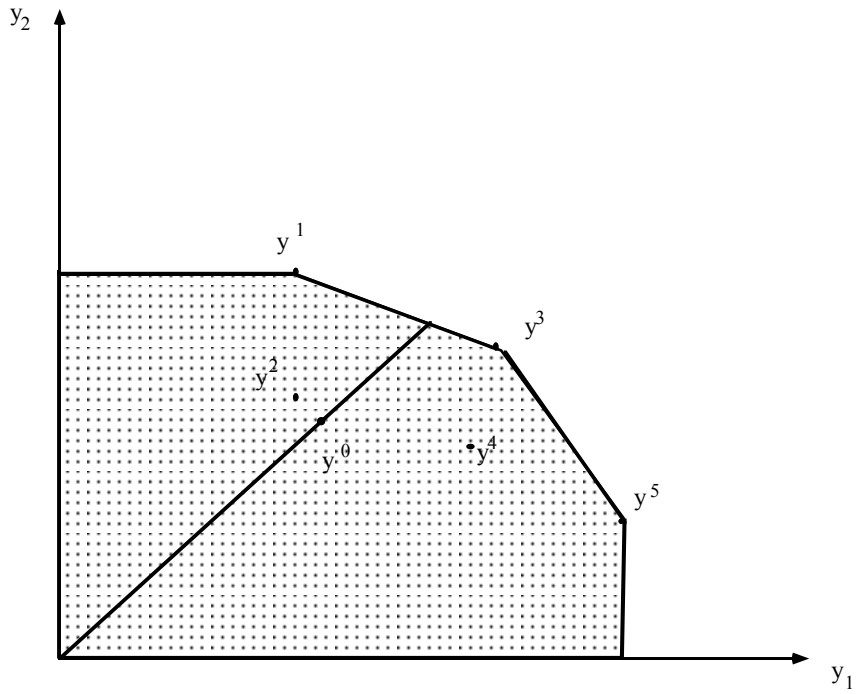
grad af efficiens vil så være at måle afstanden ud til transformationskurven på passende måde. Der er ikke noget entydigt svar på, hvordan det skal ske, men en meget anvendt metode er det såkaldte *Farrell-index,* der er den reciprokke værdi af det tal, som den aktuelle produktionsplan skal ganges op med for at nå op på transformationskurven. Der findes alternative produktivitetsindex, men dette er som sagt det mest brugte, og det rækker til vort formål.

For at beregne Farrell-indexets værdi kræves det klart nok, at man kender transformationskurvens beliggenhed. Det gør man ikke uden videre i virkelighedens verden, og det er her, at referencevirksomhederne kommer ind i billedet. Hvis det antages, at de har samme underliggende (men stadigvæk ukendte) teknologi som den aktuelle virksomhed, så har vi ihvertfald n punkter fra området på eller indenfor den sande (ukendte) transformationskurve. Antages det videre, at produktionsmulighedsområdet er konvekst, vil vi få en tilnærmet, empirisk transformationskurve. Den er bedste bud på, hvordan den virkelige ser ud, så det bedste, vi kan gøre, er at tage Farrell-index relativt til denne empiriske transformationskurve.

Hvis vektorerne af de observerede output (normeret så de svarer til 1 krones input) for referencevirksomhederne betegnes $(y_{11}, \ldots, y_{1s}), \ldots, (y_{n1}, \ldots, y_{ns})$, får vi, at området af teknologisk mulige outputvektorer bliver alle y som opfylder

$$y_r \leq \sum_{j=1}^{n} \mu_j y_{jr},\ r = 1, \ldots, s$$

for passende vægte $\mu_j \geq 0, j = 1, \ldots, n$, med $\sum_{j=1}^{n} \mu_j = 1$. Vi finder nu Farrell-indexet for den aktuelle produktion $y^0 = (y_{01}, \ldots, y_{0s})$ relativt til denne mængde; til det formål skal vi finde det største tal λ, så at λy^0 ligger i det rekonstruerede produktionsmulighedsområde (Farrell-indexet er da 1 divideret med den fundne



Figur 2

λ-værdi). Med andre ord, vi skal løse problemet

$$\begin{aligned}
&\max \lambda \\
&\text{under bibetingelserne} \\
&\sum_{j=1}^{n} \mu_j \leq 1 \\
&\sum_{j=1}^{n} \mu_j y_{jr} \geq \lambda y_{0r},\ r = 1, \ldots, s, \\
&\lambda, \mu \geq 0.
\end{aligned}$$

Bemærk ulighedstegnet ≤ 1 i den første af bibetingelserne: Vi opfatter også nulvektoren som et muligt output. De resterende bibetingelser kan også skrives som

$$\lambda y_{0r} - \sum_{j=1}^{n} \mu_j y_{jr} \leq 0,$$

og på den form er de nemmere at håndtere: Vi kigger nemlig på det duale problem,

som har formen

$$\begin{aligned}
&\min v \\
&\text{under bibetingelserne} \\
&\sum_{r=1}^{s} u_r y_{0r} \geq 1, \\
&v - \sum_{r=1}^{s} u_r y_{jr} \geq 0,\ j = 1, \ldots, n, \\
&v, u \geq 0,
\end{aligned}$$

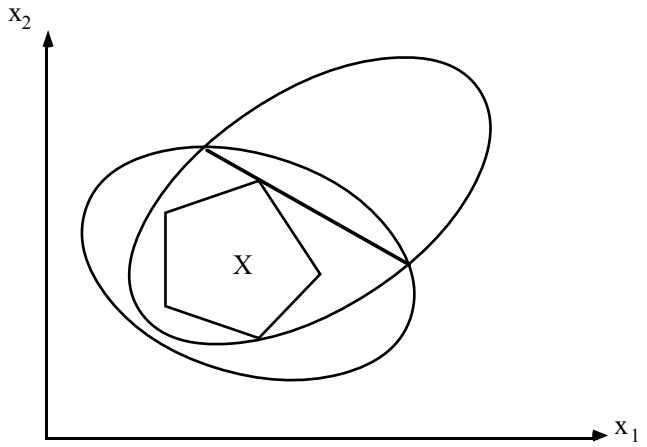
hvor vi har indført duale variable v og $u_1, \ldots, u_s$. Ulighedstegnet i den første bibetingelse kan erstattes med et lighedstegn, for hver gang vi har et sæt u'er så første bibetingelse er opfyldt med streng ulighed, såkan vi jo vælge et andet u som er mindre i alle koordinater og opfylder første bibetingelse med lighedstegn, og så vil samtlige bibetingelser stadig være opfyldt, uden at det v, vi havde at gøre med, er vokset. Det er altså nok at løse problemet

$$\begin{aligned}
&\min v \\
&\text{under bibetingelserne} \\
&\sum_{r=1}^{s} u_r y_{0r} = 1, \\
&v - \sum_{r=1}^{s} u_r y_{jr} \geq 0,\ j = 1, \ldots, n, \\
&v, u \geq 0.
\end{aligned}$$

Men dette problem er lig nøjagtig (4) i vort specialtilfælde, hvor vi har ét input og har konstant skalaafkast, så at vi kan nøjes med at se på produktioner, hvor indsatsen af dette ene input er 1.

6. Andre LP-algoritmer

Som nævnt tidligere er lineær programmering et område, hvor der stadig er intens forskning. Denne forskning går i flere retninger; dels interesserer man sig for simplex-metoden, der er den klart mest anvendte beregningsmetode. Man vil gerne have en mere grundlæggende forståelse af, hvorfor den er det, og hertil inddrages overvejelser om *beregningskompleksitet*. Men der er også i de senere år fremkommet nye metoder (igen begrundet af forskningen i kompleksitet), som har visse fortrin – indtil videre overvejende af teoretisk karakter - for simplex. Vi ser lidt på et par af disse.



Figur 1

Ellipsoid-algoritmen. Udgangspunktet er et lineært programmeringsproblem skrevet på formen

$$\min c \cdot x$$

under bibetingelserne

$$Ax \leq b$$

hvor $x \in \mathbb{R}^m$. Som sædvanlig kan LP-problemer, som er givet oprindeligt på en anden form, omskrives til denne.

Vi skal i første omgang ikke se på hele LP-problemet, men kun på det tilsyneladende langt simplere problem at finde en brugbar løsning, dvs. at finde en vektor

$$x \in X = \{x' | Ax' \leq b\}.$$

Fremgangsmåden er lettest at forklare ved at henvise til geometrien (se Figur 1). Vi har en mængde X og ønsker at finde et punkt x fra mængden. For en given vektor x er det naturligvis ret let at checke, om den tilhører mængden; det er lidt mere problematisk at finde en systematisk måde at ændre vektoren på, hvis den *ikke* tilhører mængden. Til rådighed har vi en vis særlig type mængder, nemlig ellipsoider (i figurens to dimensioner, ellipser). Vi gør derfor følgende: Start med punktet og en ellipsoid med centrum i dette punkt. Hvis punktet opfylder bibetingelserne, som definerer X, er vi færdige. Gør det ikke det, vælger vi en bibetingelse, der er overtrådt; da ligger X helt på den ene side af det halvrum, som er bestemt af vort punkt og en overtrådt bibetingelse. Vælg nu en ellipsoid, som lige netop dækker skæringen mellem den første ellipsoid og halvrummet, og gå til centrum af denne nye ellipsoid. Nu kan algoritmen fortsætte som før.

Der resterer at præcisere, hvad man egentlig skal foretage sig – samt at argumentere for, at proceduren fører til en løsning. Vi starter med to antagelser, som er ret uskyldige; den første siger, at løsningsmængden X ikke er ubegrænset, og den anden siger, at løsningsmængden indeholder en passende lille kugle (hvad der så igen medfører, at dens rumfang ikke kan være nul):

Antagelse 1. Der findes et $x_0 \in \mathbb{R}^m$ således at X er indeholdt i kuglen

$$B_R(x_0) = \{x \in \mathbb{R}^m | \|x - x_0\| \leq R\}$$

for et passende $R > 0$.

Antagelse 2. Der findes $\rho > 0$ så at $X \neq \emptyset$ medfører $B_\rho(x^*) \subset X$ for et eller andet x^*.

Metoden går nu ud på at operere med passende store "kasser", som X fra starten er lukket inde i. "Kassen" opdeles nu på en sådan måde, at løsningsmængden er i den ene af delene, og der fortsættes på samme måde, indtil man enten har en løsning eller har lukket løsningsmængden ind i en passende lille "kasse", og fra antagelserne får vi at der før eller senere må indfinde sig en løsning. Teknisk bruger vi en bestemt type "kasse":

En *ellipsoid* i $\mathbb{R}^m$ (med centrum i $\bar{x}$) er en mængde af formen

$$E = \{x | (x - \bar{x}) B^{-1} (x - \bar{x}) \leq 1\}$$

for en symmetrisk og positiv definit matrix B. Skrives denne matrix B som JJ^t for en passende matrix J (det kan man, når B er symmetrisk og positiv definit), kan vi også skrive vor ellipsoid som

$$E = \{\bar{x} + Jz | \|z\| \leq 1\},$$

hvoraf vi ser, at E er billedet af enhedskuglen $B_1(0)$ under den lineære transformation $z \mapsto \bar{x} + Jz$. Dette kan man bruge til at notere sig, at volumen af E kan skrives som

$$\text{vol}E = \text{det}J\text{vol}B_1(0) = (\text{det}B)^{1/2}\text{vol}B_1(0),$$

hvor det(J) er determinanten af matricen J (den skal man heldigvis ikke regne ud for at gennemføre algoritmen, den bruges her bare til at vise, at algoritmen gør, som den skal; overvejelser om volumen spiller en central rolle i begrundelsen for ellipsoid-algoritmen.

Fremgangsmåden i denne er nu følgende: Vi starter ved at vælge kuglen $B_R(x_0)$ som første ellipsoid, og x_0 som første vektor.

Antag at vi befinder os i trin $k \geq 0$ og har vektoren x_k, som er centrum i ellipsoiden E_k givet ved

$$E_k = \{x | (x_k - \bar{x}) B_k^{-1} (x_k - \bar{x}) \leq 1\}.$$

Vi har konstrueret E_k så $X \subset E_k$.

Hvis $x_k \in X$, er algoritmen (der som nævnt blot går ud på at finde en mulig løsning) slut. Ellers er $x_k \notin X$, og vi kan vælge en række j i A, således at uligheden $a_j \cdot x \leq c$ er overtrådt. Vi ønsker nu at bestemme et punkt i mængden

$$\{x | a_j \cdot x \leq a_j \cdot x_k\} \cap E_k,$$

som klart indeholder X. Den næste ellipsoid E_{k+1}, som skal indeholde denne mængde, laves på følgende måde:

Lad parametrene τ, δ, σ været givet som følger:

$$\tau = \frac{1}{m+1},\ \delta = \frac{m^2}{m^2-1},\ \sigma = \frac{2}{m+1}.$$

Vi opdaterer således:

$$x_{k+1} = x_k - \frac{\tau B_k a_j}{(a_j^t B a_j)^{1/2}},$$
$$B_{k+1} = \delta\left(B_k - \sigma\frac{B_k a_j a_j^t B_k}{a_j^t B_k a_j}\right),$$

hvorefter vi er klar til trin $k+1$ med ellipsoiden E_{k+1} givet ved x_{k+1} og B_{k+1}.

Det kan vises – noget som er afgørende for algoritmens funktion – at sådan som vi har opdateret, vil E_{k+1} indeholde mængden X, og der vil gælde

$$\frac{\text{vol}E_{k+1}}{\text{vol}E_k} = \left(\frac{m^2}{m^2-1}\right)^{(m-1)/2}\frac{m}{m+1}.$$

Det interessante er her, at der er et fast forhold mellem volumen af et ellipsoid og det foregående, hvilket igen betyder, at volumen af E_k må gå mod nul, når k går mod uendelig. Vi får da fra antagelse 2, at hvis der overhovedet er mulige løsninger, må vi finde en efter endelig mange trin.

Nu kan endelig mange trin jo godt være et ret stort antal. Det rigtig spændende ved ellipsoid-algoritmen er, at man kan vise, at antallet af beregningsoperationer vil være begrænset ved et *polynomium* i problemets størrelse, udtrykt ved antallet af variable og deres størrelse. Det var netop denne egenskab ved ellipsoid-algoritmen, der gjorde den til en sensation ved sin fremkomst. Simplex-algoritmen har nemlig ikke denne egenskab; se iøvrigt kapitlet om kompleksitet.

Vi er jo egentlig slet ikke færdige endnu, for vi har kun set på metoder til at finde mulige løsninger; vi søger faktisk en, som er optimal.

Det er dog ikke så svært at udbygge ellipsoid-algoritmen til at klare dette. Vi erstatter vort oprindelige LP-problem med følgende: Finde en løsning til ligningssystemet

$$\begin{aligned} Ax &\leq b \\ -yA &\leq -c \\ -y &\leq 0 \\ b\cdot y + c\cdot x &\leq 0 \end{aligned}$$

Her har vi til bibetingelserne i det oprindelige problem føjet bibetingelserne i det duale; den sidste bibetingelse siger at løsningen til det oprindelige (skrevet

som $\max -c \cdot x$) skal være $\geq$ løsningen til det duale ($\min b \cdot y$), hvilket jo kun kan indtræffe, hvis løsningen er optimal. Dermed har vi, at en løsning til ligningssystemet ovenfor giver løsning til LP-problemet (samt til det duale, der altså kommer "gratis" med ligesom ved simplex-metoden.

Eksempel 3.2

Lad os gennemgå beregningstrinene i ellipsoid eksemplet for et meget simpelt eksempel: Vi ser kun på systemet af lineære uligheder

$$\begin{aligned} x_1 + x_2 &\leq 1 \\ x_1 + 2x_2 &\geq 1 \\ 2x_1 + x_2 &\geq 1 \end{aligned}$$

hvor vi ønsker en vektor (x_1, x_2), der opfylder alle tre uligheder. Det er selvfølgelig temmelig nemt at finde en løsning med det blotte øje, men det er ikke pointen her, hvor vi er interesserede i at illustrere i teknikken i ellipsoid-algoritmen. Vi skriver ulighederne på matrixform i overensstemmelse med notationen ovenfor, således, at alle uligheder vender samme vej:

$$\begin{pmatrix} 1 & 1 \\ -1 & -2 \\ -2 & -1 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} \leq \begin{pmatrix} 1 \\ -1 \\ -1 \end{pmatrix}.$$

Vi skal have nogle passende værdier at starte med. Nulpunktet er ihvertfald let at have med at gøre, så vi sætter $x^1 = (0, 0)$; Vi skal også bruge en passende stor ellipsoid (i to dimensioner som her er det såmænd bare en ellipse). Det er let at se, at enhver løsning til ligningerne ovenfor må have norm ≤ 1 (ellers kan man se det af figur 2), så enhedskuglen er en god ellipse at starte med, og den får man ved at sætte den første matrix til en enhedsmatrix, dvs.

$$B_1 = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}.$$

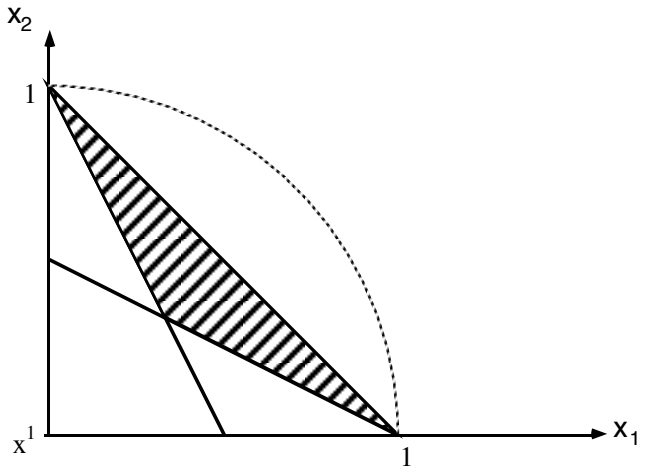
Parametrene afhænger kun af dimensionen af vort problem, her 2, så vi har

$$\tau = \frac{1}{3}, \quad \delta = \frac{4}{3}, \quad \sigma = \frac{2}{3}.$$

Så kan vi godt starte. *Første trin* består i at checke, om x^1 opfylder samtlige bibetingelser. Det gør den ikke, allerede ved den anden går det galt; også den tredie er overtrådt. Det er naturligt at bruge den "værste" af dem til opdatering; her er den manglende opfyldelse ens, så vi tager den første af dem. Så skal vi have fat i den række i matricen A, der svarer til den pågældende bibetingelse; det giver os anden række i matricen A, altså vektoren $a_2 = (-1, -2)$. Og så kan der opdateres; det nye punkt, x^2, finder vi som

$$x^2 = \begin{pmatrix} 0 \\ 0 \end{pmatrix} - \frac{1}{3} \frac{\begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \begin{pmatrix} -1 \\ -2 \end{pmatrix}}{\sqrt{\begin{pmatrix} -1 & -2 \end{pmatrix} \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \begin{pmatrix} -1 \\ -2 \end{pmatrix}}}.$$

Eksempel 3.2, fortsat



Figur 2

Det ser måske ud af meget, men matrixprodukterne er jo ret simple, når enhedsmatricen ganges på, der hvor den står; tælleren bliver til kvadratroden af $1 \cdot 1 + 2 \cdot 2 = 5$, og alt i alt får vi

$$x^2 = \begin{pmatrix} \frac{1}{3\sqrt{5}} \\ \frac{2}{3\sqrt{5}} \end{pmatrix}.$$

Opdatering af B_1 sker ligeledes ved matrixprodukter; vi skal lave

$$B_2 = \frac{4}{3}\left[\begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} - \frac{2}{3}\frac{1}{5}\begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}\begin{pmatrix} -1 \\ -2 \end{pmatrix}\begin{pmatrix} -1 & -2 \end{pmatrix}\begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}\right].$$

Igen ser det værre ud, end det er. Hvis vi starter indefra i det lange matrixprodukt, har vi først

$$\begin{pmatrix} -1 \\ -2 \end{pmatrix}\begin{pmatrix} -1 & -2 \end{pmatrix} = \begin{pmatrix} 1 & 2 \\ 2 & 4 \end{pmatrix};$$

enhedsmatricerne foran og bagved kan smides væk, så vi har

$$B_2 = \frac{4}{3}\left[\begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} - \begin{pmatrix} \frac{2}{15} & \frac{4}{15} \\ \frac{4}{15} & \frac{8}{15} \end{pmatrix}\right] = \begin{pmatrix} \frac{52}{45} & -\frac{16}{45} \\ -\frac{16}{45} & \frac{28}{45} \end{pmatrix}.$$

Det var så første trin. Ser vi nærmere på, hvad der skete, så er vi nu flyttet fra nulpunktet op til x^2, som er væsentlig nærmere på den linie, der bestemmer anden bibetingelse (men ikke helt op på den). Den oprindelige kugle er blevet til en ellipse (som vi ikke tegner.)

Vi er nu klar til næste trin. Vi checker bibetingelserne og ser, at den stadig er gal med såvel den anden som den tredie. Men den tredie har det værst, så det vil være oplagt at bruge denne.

Vi overlader eksemplets videre skæbne til en eventuel udholdende læser, der vil kunne bringe x^k nærmere til en løsning; man har allerede her en fornemmelse af, at det godt kan tage et stykke tid.

Karmarkar's algoritme. Ellipsoid-algoritmen i forrige afsnit udmærker sig ved teoretisk pæne egenskaber, men i praksis er den ikke særlig hurtig. Den er god i de

værst tænkelige tilfælde, mindre fremragende i de øvrige. En anden ny algoritme, foreslået af Karmarkar i 1984, har også de teoretisk set pæne egenskaber, og den har indtil videre været noget mere lovende i praksis.

Karmarkar's metode tager udgangspunkt i et LP-program af følgende form:

$$\begin{aligned}
&\min c \cdot x \\
&\text{under bibetingelserne} \\
&Ax = 0 \\
&x \geq 0, \sum_{j=1}^{n} x_j = 1.
\end{aligned}$$

Bemærk, at vi her søger en løsning til LP-problemet i simplexen bestående af alle ikke-negative vektorer med koordinatsum 1. Der gøres indledningsvis nogle antagelser:

Antagelse 1. Matricen A, som er $(m \times n)$, har rang m, og vektoren $e = (1, 1, \ldots, 1)$ opfylder betingelsen $Ae = 0$.

Antagelse 2. Den optimale værdi af $c \cdot x$ er 0.

Det kan umiddelbart se temmelig specielt ud, at LP-problemet er givet med den ekstra betingelse, at x kun kan variere i simplexen. Det er dog ikke nogen særlig væsentlig restriktion; går vi over til det duale problem, får vi

$$\begin{aligned}
&\max z \\
&\text{under bibetingelserne} \\
&yA + ze \leq c \\
&y, z \geq 0.
\end{aligned}$$

Da den optimale værdi af dette program ifølge antagelse 2 er 0, søger vi egentlig blot y således at $yA \leq c$, og dette er et problem, der svarer til, hvad vi beskæftigede os med ved ellipsoid-metoden.

Antagelserne er denne gang ret restriktive, specielt med hensyn til optimalværdien. Det vender vi tilbage til.

Karmarkar's metode benytter sig af *projektive transformationer:* Lad $\bar{x}$ være et punkt i i det indre af simplexet (så at $\bar{x}_i > 0$ for alle i), som opfylder bibetingelsen $Ax = 0$, men som er forskellig fra barycentret i simplexet; vi søger en afbildning, som sender simplexets hjørner i sig selv og får $\bar{x}$ til at gå i barycentret,

$$\bar{x} \mapsto \left(\frac{1}{n}, \ldots, \frac{1}{n} \right).$$

Det kan ikke være en lineær afbildning, for når hjørnerne skal sendes i sig selv, bliver den identisk og kan derfor ikke flytte $\bar{x}$. Men vi kan lave en, der næsten er

lineær: Når $\bar{x}$ skal sendes i en vektor med lige store koordinater, er det fristende at prøve afbildningen

$$x \mapsto \left(\frac{x_1}{\bar{x}_1}, \ldots, \frac{x_n}{\bar{x}_n} \right),$$

som dog ikke sender simplexet i sig selv; det kan vi imidlertid opnå ved at normere, dvs. dividere med summen af koordinaterne. For at få en brugbar notation indfører vi $(n \times n)$-matricen $\bar{X}$ med $\bar{x}$ i diagonalen, 0 udenfor. Vor afbildning skal da være

$$x \mapsto \hat{x} = \frac{\bar{X}^{-1}x}{e^t \bar{X}^{-1}x},$$

som har invers givet ved

$$x = \frac{\bar{X}\hat{x}}{e^t \bar{X}\hat{x}}.$$

Hvis vi nu indsætter dette sidste udtryk i vort oprindelige LP-problem, bliver det til:

$$\begin{aligned} &\min \frac{c^t \bar{X}\hat{x}}{e^t \bar{X}\hat{x}} \\ &\text{under bibetingelserne} \\ &A\bar{X}\hat{x} = 0, \\ &\hat{x} \geq 0, \ \sum_{i=1}^{n} \hat{x}_i = 1. \end{aligned}$$

Da vi leder efter en løsning med kriterieværdien 0, må tælleren i kriteriefunktionen være 0, og vi kan derfor glemme nævneren. Skriver vi $\hat{A} = A\bar{X}$, $\hat{c} = \bar{X}c$, får vi LP-problemet

$$\begin{aligned} &\min \hat{c} \cdot \hat{x} \\ &\text{under bibetingelserne} \\ &\hat{A}\hat{x} = 0, \\ &\hat{x} = 0, \ \sum_{i=1}^{n} \hat{x}_i = 1. \end{aligned}$$

Dette er et problem magen til det vi startede med, og vi har en mulig løsning, nemlig barycentret (som før transformationen var vor vektor $\bar{x}$).

Vi har dermed gennemgået, hvorledes problemet opdateres i hvert trin. Vi mangler kun et kriterie for, hvorledes der vælges nyt punkt i det indre af simplexet. Men det er ret ligetil: Hvis barycentret ikke er en optimal løsning, er der en retning væk fra det, hvor kriteriefunktionen bliver mindre. En sådan retning kan findes ved at projicere $\hat{c}$ ned på simplexet, hvilket giver os vektoren

$$\hat{d} = -(I - B^t(BB^t)^{-1}B)\hat{c},$$

hvor

$$B = \begin{pmatrix} A \\ e \end{pmatrix}.$$

Vi flytter os nu et lille stykke ρ i $\hat{d}$-retningen, hvorved vi får det nye punkt

$$\hat{x}(\rho) = \frac{e}{n} + \rho \frac{\hat{d}}{\|\hat{d}\|},$$

og dermed har vi en ny mulig løsning udenfor barycentret, og vi kan fortsætte. Størrelsen ρ er fast; den skal være så tilpas lille, at man ikke risikerer at ramme randen af simplexet; hvis man vælger $\rho < [n(n-1)]^{-1/2}$. er man på den sikre side.

Eksempel 3.3

Her er et eksempel på, hvordan Karmarkar's algoritme kører: Vi betragter det lineære programmeringsproblem

$$\max 2x_1 + x_2$$

under bibetingelserne

$$x_1 - x_2 = 0$$

$$x_1 + x_2 + x_3 = 1, x_1, x_2, x_3 \geq 0$$

eller, skrevet på en form, der svarer til opstillingen ovenfor,

$$\max \begin{pmatrix} 2 & 1 & 0 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix}$$

under bibetingelserne

$$\begin{pmatrix} 1 & -1 & 0 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix}$$

$$x_1 + x_2 + x_3 = 1, x_1, x_2, x_3 \geq 0$$

Det ses, at x_3 ikke spiller nogen særlig rolle i problem; på den anden side ser vi, at takket være formuleringen her opfylder problemet de lidt mærkelige generelle antagelser: for det første er diagonalvektoren løsning til bibetingelsen, og for det andet er den optimale værdi af problemet lig nul (uden x_3 er kriterieværdien altid ikke-negativ, og med x_3 alene er den nul. Dette viser lidt om, at ved at tilføje variable kan man få de noget specielle udgangsantagelser opfyldt; det er der ikke noget specielt i, for det var også hvad vi gjorde i simplex-metoden ved indføjelse af slackvariable.

Eksempel 3.3, fortsat

Hvorom alting er, her er vort problem, og vi går straks igang. Først checker vi, om barycentret

$$\left(\frac{1}{3}, \frac{1}{3}, \frac{1}{3}\right)$$

løser problemet, simpelthen ved at udregne kriteriets værdi på denne vektor. Det bliver

$$2 \cdot \frac{1}{3} + 1 \cdot \frac{1}{3} + 0 \cdot \frac{1}{3} > 0,$$

så vi er ikke i optimalpunktet (hvor kriterieværdien som bekendt er 0). Altså må der opdateres. Først finder vi retningen ud til et nyt punkt (idet c-vektoren projiceres ned på planet af vektorer med koordinatsum 0, og der skiftes fortegn så det bliver en retning, hvor kriteriet aftager), hvilket giver os

$$d = -\left[I - \begin{pmatrix} 1 & 1 \\ -1 & 1 \\ 0 & 1 \end{pmatrix} \left(\begin{pmatrix} 1 & -1 & 0 \\ 1 & 1 & 1 \end{pmatrix} \begin{pmatrix} 1 & 1 \\ -1 & 1 \\ 0 & 1 \end{pmatrix} \right)^{-1} \begin{pmatrix} 1 & -1 & 0 \\ 1 & 1 & 1 \end{pmatrix} \right] \begin{pmatrix} 2 \\ 1 \\ 0 \end{pmatrix}.$$

Det ser værre ud, end det er. Vi tager først det matrixprodukt, der skal inverteres, nemlig

$$\left(\begin{pmatrix} 1 & -1 & 0 \\ 1 & 1 & 1 \end{pmatrix} \begin{pmatrix} 1 & 1 \\ -1 & 1 \\ 0 & 1 \end{pmatrix} \right)^{-1} = \begin{pmatrix} 2 & 0 \\ 0 & 3 \end{pmatrix}^{-1} = \begin{pmatrix} \frac{1}{2} & 0 \\ 0 & \frac{1}{3} \end{pmatrix},$$

så det var ret ledt overstået. Så ganger vi denne fra venstre og fra højre med de respektive matricer til

$$\begin{pmatrix} 1 & 1 \\ -1 & 1 \\ 0 & 1 \end{pmatrix} \begin{pmatrix} \frac{1}{2} & 0 \\ 0 & \frac{1}{3} \end{pmatrix} \begin{pmatrix} 1 & -1 & 0 \\ 1 & 1 & 1 \end{pmatrix} = \begin{pmatrix} \frac{5}{6} & -\frac{1}{6} & \frac{1}{3} \\ -\frac{1}{6} & \frac{5}{6} & \frac{1}{3} \\ \frac{1}{3} & \frac{1}{3} & \frac{1}{3} \end{pmatrix}.$$

Vi trækker så fra enhedsmatricen og ganger med den sidste søjlevektor; ialt fås:

$$d^1 = \begin{pmatrix} \frac{1}{6} & \frac{1}{6} & -\frac{1}{3} \\ \frac{1}{6} & \frac{1}{6} & -\frac{1}{3} \\ -\frac{1}{3} & -\frac{1}{3} & \frac{2}{3} \end{pmatrix} \begin{pmatrix} 2 \\ 1 \\ 0 \end{pmatrix} = \begin{pmatrix} \frac{1}{2} \\ \frac{1}{2} \\ -1 \end{pmatrix}.$$

Vektoren har norm

$$\sqrt{2(1/2)^2 + 1} = \sqrt{\frac{3}{2}},$$

og skridtlængden ρ skal være mindre end $1/\sqrt{6}$, f.eks. $1/2\sqrt{6}$, således at vi flytter os fra barycentret til

$$\begin{aligned} \bar{x}^1 &= \begin{pmatrix} \frac{1}{3} \\ \frac{1}{3} \\ \frac{1}{3} \end{pmatrix} - \frac{1}{2\sqrt{6}} \sqrt{\frac{2}{3}} \begin{pmatrix} \frac{1}{2} \\ \frac{1}{2} \\ -1 \end{pmatrix} \\ &= \begin{pmatrix} \frac{1}{3} \\ \frac{1}{3} \\ \frac{1}{3} \end{pmatrix} - \begin{pmatrix} \frac{1}{12} \\ \frac{1}{12} \\ -\frac{1}{6} \end{pmatrix} = \begin{pmatrix} \frac{3}{12} \\ \frac{3}{12} \\ \frac{3}{6} \end{pmatrix} = \begin{pmatrix} \frac{1}{4} \\ \frac{1}{4} \\ \frac{1}{2} \end{pmatrix}. \end{aligned}$$

Eksempel 3.3, fortsat

Nu har vi det nye indre punkt i matricen, og så kan vi opdatere vort problem: Den nye c-vektor (i transponeret udgave) bliver

$$\begin{pmatrix} 2 & 1 & 0 \end{pmatrix} \begin{pmatrix} \frac{1}{4} & 0 & 0 \\ 0 & \frac{1}{4} & 0 \\ 0 & 0 & \frac{1}{2} \end{pmatrix} = \begin{pmatrix} \frac{1}{2} & \frac{1}{4} & 0 \end{pmatrix}$$

og den nye A bliver til

$$\begin{pmatrix} 1 & -1 & 0 \end{pmatrix} \begin{pmatrix} \frac{1}{4} & 0 & 0 \\ 0 & \frac{1}{4} & 0 \\ 0 & 0 & \frac{1}{2} \end{pmatrix} = \begin{pmatrix} \frac{1}{4} & -\frac{1}{4} & 0 \end{pmatrix}.$$

Så er vi klar til næste skridt.

Vi starter med et check af, om barycentret løser vort nye problem, dvs. om dets kriterieværdi er 0. Det er vi stadig ikke så heldige med, vi har

$$\frac{1}{2} \cdot \frac{1}{3} + \frac{1}{4} \cdot \frac{1}{3} = \frac{1}{6} + \frac{1}{12} = \frac{1}{4} > 0,$$

så vi kommer til at fortsætte. Som ved forrige eksempel vil vi overlade fortsættelsen til den energiske læser.

Som nævnt har Karmarkar's algoritme (der siden fremkomsten er blevet til en hel familie af såkaldte indre-punkt-algoritmer) vist sig bedre i praksis end ellipsoid-algoritmen. Men denne deler Karmarkar's metode den pæne egenskab, at antallet af beregninger er polynomial i problemets størrelse.

I praktiske anvendelser har man det særlige problem, at vi har antaget noget ganske bestemt om løsningen, nemlig at kriterieværdien var 0. Det er naturligvis mindre afgørende, om den er 0 er et hvilketsomhelst andet tal, problemet er, at den optimale kriterieværdi forudsættes kendt før algoritmen starter; det er jo som oftest fordi man ikke kender den, at man gennemfører beregningerne. Det er dog ikke helt håbløst; hvis man fra starten har en øvre og nedre grænse, kan man tilpasse algoritmen, så at den virker.

7. Opgaver

1. Find en (tilnærmet) løsning til systemet af uligheder

$$\begin{aligned} x_1 + 3x_2 + 3x_3 &\leq 14 \\ 2x_1 + 3x_2 + x_3 &\leq 12 \\ x_1 + x_2 + x_3 &\geq 6 \end{aligned}$$

ved brug af ellipsoid-metoden (det er nemt at finde en løsning uden at regne, men

det er altså ikke pointen lige i denne opgave). Der ønskes mindst 3 iterationer (hvis ikke der er fundet en løsning inden da).

2. Betragt det lineære programmeringsproblem

$$\begin{aligned}
&\min 10y_1 + 14y_2 + 15y_3 \\
&\text{under bibetingelserne} \\
&y_1 + 7y_2 + 2y_3 \geq 3 \\
&2y_1 + 4y_2 \geq 2 \\
&2y_1 + 8y_3 \geq 1 \\
&y_1, y_2, y_3 \geq 0
\end{aligned}$$

Løs problemet ved dual simplex (idet man maximerer $-10y_1 - 14y_2 - 15y_3$ under bibetingelserne ganget igennem med -1. Sammenlign med den løsningsmetode, der går ud på at formulere det duale problem og løse dette på sædvanlig måde.

3*. Løs det lineære programmeringsproblem

$$\begin{aligned}
&\max 2x_1 + x_2 \\
&\text{under bibetingelserne} \\
&x_1 - x_2 = 0 \\
&x_1, x_2, x_3 \geq 0, x_1 + x_2 + x_3 = 1
\end{aligned}$$

ved Karmarkar's metode.

8. Litteratur

Fremstillingen af DEA følger Charnes, Cooper og Rhodes (1978). Den kortfattede beskrivelse af Farrell-indexet yder ikke dette emne fuld retfærdighed, men det falder udenfor vor emnekreds; der kan henvises til Färe, Grosskopf og Lovell (1984).

At der stadig er liv i forskningen relateret til lineær programmering, kan man overbevise sig om ved at læse i Lagarias og Todd (1990). Ellipsoid-metoden er egentlig et biprodukt af udviklingen af metoder indenfor ikke-lineær programmering (Shor 1970). Det afgørende gennembrud kom med Khachian (1979), der viste, at ellipsoid-algoritmen er polynomial.

KAPITEL 4

Ikke-lineær programmering

1. Indledning

Hvis kriteriefunktionen eller bibetingelserne ikke er lineære, har vi i princippet et problem, der ikke kan løses ved lineær programmering. Ofte vil man kunne klare sig ved at tilnærme det rigtige problem med et, som er lineært, men også dette kan efter omstændighederne vise sig at glippe. Vi skal se lidt nærmere på, hvad man så kan gøre.

Den konkrete metode vil afhænge ret meget af, hvorledes kriteriefunktionen (eller bibetingelserne) ser ud. Der er ikke gode metoder, som lader sig bruge i alle situationer. Vi ser nedenfor på udvalgte tilfælde; dermed kommer vi langtfra ind på alle typer af optimeringsmetoder for ikke-lineære problemer, men det kan give en smagsprøve på, hvad man kan løse i praksis.

Den første metode, kvadratisk programmering, benyttes når kriteriefunktionen indeholder andengradsled (en variabel i anden potens eller produkter af to variable). Tricket er her at reducere problemet til et, hvor der bare skal løses lineære ligninger; det sker ved at bruge de klassiske betingelser for optimum ("differentiere og sætte lig nul"). Der er selvfølgelig nogle praktiske detaljer, der skal styres, for at denne metode fungerer, og det er egentlig blot det, som udgør den konkrete metode for kvadratisk programmering.

Vi nøjes med en enkelt anden metode, nemlig den såkaldte konvekse simplex. Denne metode kan bruges, når kriteriefunktionen er en konveks funktion (ihvertfald til minimeringsproblemer; vi ser på maximering, så vi forudsætter kriteriefunktionen konkav , det kommer ud på det samme). Som overalt i dette kapitel kræver vi, at bibetingelserne er givet ved lineære uligheder. Metoden kan anvendes på væsentlig flere problemer end den kvadratiske programmering, men den er til gengæld også væsentlig mere omstændelig, idet der kan forventes mange iterationer og temmelig besværlige regninger i hver iteration.

2. Kvadratisk programmering: Wolfe's algoritme

Vi vil starte vor diskussion af udvidelser af metoderne til ikke-lineære problemer

med en gennemgang af optimeringsmetoder for kvadratisk kriteriefunktion, noget der ikke overraskende går under navnet *kvadratisk programmering*. Den grundlæggende ide – her som i mangt og meget senere hen – er at føre problemet tilbage til noget velkendt, hvilket på vort nuværende niveau vil sige at omformulere det til noget lineært.

Et eksempel vil vise, hvordan man kan gøre det:

Eksempel 4.1

Lad os til en begyndelse se på det optimeringsproblem, som går ud på at minimere funktionen

$$x^2 + y^2$$

under bibetingelsen

$$2x + y = 1$$

samt ikke-negativitetsbetingelserne $x, y \geq 0$. Vi vil ikke fordybe os i, hvorledes dette problem eventuelt kan være opstået af en eller anden økonomisk sammenhæng, men blot se på løsning af det.

Standard-teknikken fra optimeringsmodellerne i økonomisk teori er at lave Lagrange-funktionen

$$L(x, y, \lambda) = x^2 + y^2 + \lambda(1 - 2x - y),$$

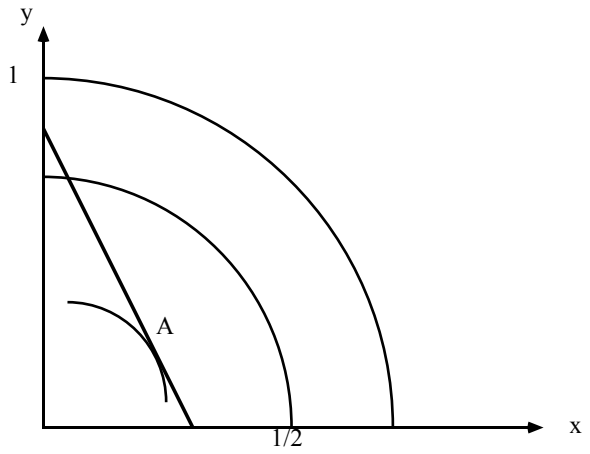
differentiere med hensyn til de variable i denne, og derefter løse de ligninger, der kommer ud af det. Som bekendt er dette nødvendige betingelser – hvis (x^0, y^0) løser minimeringsproblemet, må de også være løsning til ligningssystemet.

Ligningssystemet får formen

$$\begin{aligned} 2x - 2\lambda &= 0 \\ 2y - \lambda &= 0 \\ 2x + y &= 1 \end{aligned}$$

med løsning

$$(x^0, y^0 = \left(\frac{2}{5}, \frac{1}{5}\right) \text{ og } \lambda = \frac{2}{5}.$$



Figur 1

Eksempel 4.1, fortsat

Geometrien i problemet, vist i figur 1, overbeviser os hurtigt om, at vi har at gøre med et minimum: Vi skal befinde os på den rette linie (bibetingelsen) og samtidig søge den mindst mulige cirkelbue; det giver den løsning, vi fandt ovenfor. Vi noterer os også, at vi fik ikke-negativitetsbetingelserne overholdt nærmest som en foræring, for vi tog ikke hensyn til det undervejs.

Sådanne "gaver" kan man ikke regne med i almindelighed. Derfor må man sørge for at inddrage noget systematik i sin metode, så at man ikke efterfølgende skal lave de hele om på grund af forkerte fortegn. Man er også nødt til at overveje, om ens løsning er et maximum eller et minimum.

I et kvadratisk programmeringsproblem er kriteriefunktionen kvadratisk, mens bibetingelser som tidligere er lineære. Det har altså formen

$$\begin{aligned} &\max c \cdot x + x^t D x \\ &\text{under bibetingelserne} \\ &Ax = b \\ &x \geq 0. \end{aligned}$$

Her er x en vektor med n koordinater (efter omstændighederne opfattet som vektor eller som søjlematrix).

Matricen D, som har dimensionen $(n \times n)$, antages at D være *negativ semi-definit,* hvilket betyder at der for en hvilkensomhelst n-vektor u vil gælde, at

$$u^t D u = \sum_{i=1}^{n} \sum_{j=1}^{n} d_{ij} u_i u_j \leq 0.$$

Denne betingelse er selvfølgelig ikke let at checke umiddelbart, så man har brug for andre metoder til det, og sådanne findes. Man tager alle principale underdeterminante af orden k, og så skal determinanternes fortegn s kifte, startende med noget ikke-positivt. Dette check er heller ikke særlig forbrugervenligt, men i simple tilfælde går det som regel.

Antagelsen om at D er negativ semidefinit skal alene bruges til at sikre, at det, man finder, rent faktisk er et maximum og ikke et minimum. Betingelsen er ækvivalent med, at den kvadratiske kriteriefunktion f er *konkav,* (således at $f(\lambda x + (1-\lambda)y) \geq \lambda x + (1-\lambda) f(y)$ for alle x, y i f's definitionsmængde og $\lambda \in [0,1]$. Når f er konkav, vil et lokalt maximum også være et globalt maximum, og dette sikrer, at metoden faktisk beregner et maximum.

Da bibetingelserne er lineære, er det fristende at bruge teknikken fra lineær programmering (det vil i praksis sige simplex-metoden) til at finde mulige løsninger. Vi skal så supplere med en metode til at finde en optimal løsning; her må vi finde på noget nyt, for simplex-metoden giver ikke nogen brugbar anvisning i vor situation.

Vi skal se på en metode, som skyldes Wolfe (1959). Ideen er at bruge simplex-metoden til at finde en mulig løsning, der opfylder Kuhn-Tucker betingelserne. Det betyder, at vi leder efter x, λ og μ, som løser ligningssystemet

$$\begin{aligned} & c + 2Dx - A^t\lambda + \mu = 0 \\ & \sum_{j=1}^{n} \mu_j x_j = 0, \\ & \mu \geq 0 \end{aligned}$$

hvor $\lambda = (\lambda_1, \ldots, \lambda_m)$ er en vektor af Lagrange-multiplikatorer (hørende til hver af ligningerne i systemet $Ax = b$ – da dette er givet ved lighedstegn, er der ingen fortegnsrestriktioner på λ'erne), og $\mu = (\mu_1, \ldots, \mu_n)$ er Lagrange-multiplikatorer hørende til ikke-negativitetsbetingelserne for x.

Dette system kan også skrives som

$$\begin{aligned} 2Dx - A^t\lambda + \mu &= -c \\ \mu &\geq 0 \\ \mu_j x_j &= 0,\ j = 1, \ldots, n \end{aligned}$$

Føjes hertil de oprindelige bibetingelser

$$Ax = b,$$

har vi et ligningssystem, der næsten svarer til dem, som vi kan anvende simplex-metoden på; eneste problem er ligningerne $\mu_j x_j = 0$, hvor både variablene x_j og μ_j indgår, altså kvadratiske led, og det viser sig ikke at volde de helt store kvaler. Vi vil derfor at klargøre til simplex-metoden, hvilket betyder, at vi skal have ikke-negativitetsbetingelser på alle variable (dvs. også λ_i).

Vi skriver først vore variable λ_i som

$$\lambda_i = \lambda_i' - \lambda_0',\ \lambda_i', \lambda_0' \geq 0$$

for hvert i, eller. på matrixform,

$$\lambda = \lambda' - \lambda_0' e,$$

hvor e er en vektoren bestående af m et-taller. Indsættes dette, får vi ligningssystemet

$$\begin{aligned} Ax &= b \\ 2Dx - A^t\lambda' + \lambda_0' A^t e + \mu &= -c \\ x, \lambda', \lambda_0', \mu &\geq 0, \\ \mu_j x_j &= 0,\ j = 1, \ldots, n. \end{aligned}$$

Vi noterer os nu følgende: Hvis der findes en mulig løsning til dette system, da findes der en mulig *basis*-løsning til ligningssystemet minus ligningerne i den sidste linie, dvs. systemet

$$\begin{aligned} Ax &= b \\ 2Dx - A^t\lambda' + \lambda'_0 A^t e + \mu &= -c \\ x, \lambda', \lambda'_0, \mu &\geq 0. \end{aligned}$$

For at se det lader vi $\bar{x}, \bar{\lambda}', \bar{\lambda}'_0, \bar{\mu}$ være en løsning til det store system, hvis vi trækker $\lambda^* = \min\{\lambda'_0, \lambda'_1, \ldots, \lambda'_m\}$ fra alle λ'_i, herunder λ'_0, har vi stadig en løsning (på grund af vor konstruktion af variablene λ'_i), men der kan højest være m af dem, som er forskellige fra 0. Af de resterende $2n$ variable har vi fra nederste betingelse, at der højest er n forskellige fra 0. Ialt er der altså højest $n + m$ ikke-nul elementer i løsningen, som dermed må være en basisløsning i det nederste system.

Dermed er vi klar til at gennemføre Wolfe's algoritme. Den går ud på at bruge simplex-algoritmen til at finde en løsning til ligningssystemet ovenfor, noget som simplex er velegnet til. I princippet er alt, hvad vi skal gøre, at opstille et tableau svarende til systemet, dvs. tableauet

b	A	0	0	0
$-c$	$2D$	$-A^t$	$A^t e$	I

Her skal vi blot have os en basisløsning, som har den egenskab, at hvis der er et x_j i basisløsningen med værdi > 0, så er μ_j ikke i løsningen. Det er, som man kan se, et problem om at finde en eller anden basis, noget vi erindrer fra diskussionen af simplex. Standardfremgangsmåden er her, at man tilføjer kunstige slackvariable, som man derefter forsøger at få ud (f.eks. ved at maximere en kunstig kriteriefunktion med koefficienter -1 for de kunstige variable og 0 ellers.

Brugt uden forbehold vil denne metode føre til, at vi føjer $m + n$ nye slackvariable på, en pæn udvidelse af det i forvejen ret store tableau. Ofte behøver man dog ikke så mange, for man har ofte umiddelbart en nogle gode kandidater til en basisløsning ihvertfald i de oprindelige bibetingelser, og dem kan man så supplere op. Det ser også ud til, at der er lovende kandidater at hente fra μ_j-variablene, men her er sagen lidt mere tricky – man må dels ikke bruge et μ_j hvis man allerede har taget x_j, og der er desuden et problem med minus-tegnet i vektoren c's komponenter, som kan føre til, at vi ikke kan bruge den.

For at angive en sikker metode til, hvor lidt man kan nøjes med som kunstige variable, plejer man at skrive hele Wolfe-algoritmen lidt mere detaljeret op:

Trin 1. Brug simplex-metoden til at finde en mulig basisløsning i ligningssystemet $Ax = b$. Vi skriver denne som en m-vektor $x_B \geq 0$ (resten er 0) og lader B være de tilhørende søjler i A, dvs. $Bx_B = b$; tilsvarende lader vi D_B

den $(n \times m)$ matrix, der fremkommer ved at vælge de søjler fra D, der svarer til basis. At finde en løsning til dette system er en forholdsvis lille opgave (ofte har man endda en færdig løsning, hvis problemet var givet ved ulighedstegn). Bemærk, at hvis der ikke er nogen løsning i denne fase, da er der heller ingen løsning til det oprindelige kvadratiske programmeringsproblem, og så kan man jo stoppe allerede her.

Trin 2. Vi udregner størrelsen nu $\hat{u} = -c - 2D_B x_B$. Denne størrelse er egentlig kun sjov fordi vi leder efter n yderligere basisvariable til vort store problem, og hvis vi holder x'erne fast på x_B og kun leder i de øvrige variable, bliver højresiden lig $\hat{u}$. Når den pågældende variabel er negativ, tilføjer vi en kunstig variabel med koefficient -1 (så passer pengene når vi ganger igennem med -1). Ellers tilføjer vi en med koefficient 1. Ialt sker der altså det, at vi vi tilføjer en ny vektor af variable $u = (u_1, \ldots, u_n)$, som føjes til Kuhn-Tucker ligningerne med same fortegn som $\hat{u}$. Formelt definerer vi koefficienterne Δ_j ved

$$\Delta_j = \begin{cases} +1 & \text{hvis } \hat{u}_j \geq 0, \\ -1 & \text{hvis } \hat{u}_j < 0, \end{cases}$$

og lader Δ være diagonalmatricen med Δ_j på (j, j)'te plads. Vi interesserer os nu for ligningssystemet

$$\begin{aligned} Ax &= b, \\ 2Dx - A^t\lambda' + A^t e\lambda_0' + \mu + \Delta u &= -c, \\ x, \lambda', \lambda_0', \mu, u &\geq 0. \end{aligned}$$

Opstillet på tableauform bliver dette til:

b	A	0	0	0	0
$-c$	$2D$	$-A^t$	$A^t e$	I	$\triangle$

Til brug i simplex-algoritmen ganges der igennem med -1 i de rækker j for hvilke $c_j > 0$.

Trin 3. Vi har en værdi af x fundet i trin 1, og vi sætter de øvrige variable til følgende:

$$\lambda' = 0,\ \lambda_0 = 0,\ \mu = 0,\ u_j = \Delta_j \hat{u}_j,\ j = 1, \ldots, n.$$

Der kan her højest være $m + n$ variable forskellig fra 0 (og i så fald positive); vi har en løsning til ligningssystemet beskrevet i trin 2. Det er endda en basisløsning, idet matricen af tilsvarende søjler, skrevet på blokform som

$$\begin{pmatrix} B & 0 \\ 2D_B & \Delta \end{pmatrix}$$

er regulær; den har nemlig invers

$$\begin{pmatrix} B^{-1} & 0 \\ -2\Delta D_B B^{-1} & \Delta \end{pmatrix}$$

(det checkes let ved at regne efter og bruge, at $\Delta\Delta = I$.

Trin 4. Vi har nu en basisløsning og kan bruge simplex, idet vi minimerer

$$z = u_1 + \ldots + u_n;$$

der er blot én lille modifikation, nemlig at vi ikke må have x_j og μ_j inde i basis samtidig (med mindre den ene har vægt 0 i forvejen eller får vægt 0, når den kommer ind). Da vor startbasis har egenskaben, skal vi blot sikre os, at vi ikke inddrager x_j, hvis vi allerede har μ_j (og denne ikke dermed forlader basis), eller omvendt. Dette kaldes for "Wolfe-simplex"; vi stopper som vanlig, når ingen variabel i c-rækken har negativ koefficient. Af den fundne løsning bruger vi kun x-komponenten; den er løsningen til det oprindelige kvadratiske programmeringsproblem.

Da algoritmen ovenfor er baseret på simplex, ved vi, at den stopper efter endelig mange skridt. At det faktisk – som anført ovenfor – er løsningen til det oprindelige problem, er mindre klart. Faktisk kræver det et bevis, som vi dog ikke vil gå ind på her; det kan findes f.eks. i Simmons [1975], kap 6.

Beregningerne i forbindelse med et kvadratisk programmeringsproblem kan udføres ved hjælp af et tableau, der dog nu bliver noget større, end problemet umiddelbart antyder.

Eksempel 4.2

Lad os se lidt nærmere på problemet:

$$\max 3x_1 + 4x_2 - x_1^2 + 2x_1x_2 - 2x_2^2$$

under bibetingelserne

$$x_1 + 2x_2 \leq 7$$
$$-x_1 + 2x_2 \leq 4$$
$$x_1, x_2 \geq 0.$$

Det er klart nok at vi har et kvadratisk programmeringsproblem, kriteriet er en kvadratisk funktion i de to variable x_1 og x_2. For at kunne genkende vor tidligere formalisme, ser vi at det kvadratiske led også kan skrives som

$$x^t \tilde{D} x$$

med

$$\tilde{D} = \begin{pmatrix} -1 & 1 \\ 1 & -2 \end{pmatrix}.$$

Eksempel 4.2, fortsat

Grunden til, at vi ikke kalder denne matrix for D, er at vi ikke har den færdige formulering endnu; bibetingelserne er jo ikke givet ved lighedstegn, som de skulle være, så vi bliver nødt til at tilføje slackvariable x_3 og x_4. Dermed er vi klar til at bruge Wolfe's algoritme, idet vi nu har

$$A = \begin{pmatrix} 1 & 2 & 1 & 0 \\ -1 & 2 & 0 & 1 \end{pmatrix}, \quad b = \begin{pmatrix} 7 \\ 4 \end{pmatrix}$$

$$D = \begin{pmatrix} -1 & 1 & 0 & 0 \\ 1 & -2 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix}, \quad c = \begin{pmatrix} 3 \\ 4 \\ 0 \\ 0 \end{pmatrix}$$

Vi har, at

$$A^t e = \begin{pmatrix} 1 & -1 \\ 2 & 2 \\ 1 & 0 \\ 0 & 1 \end{pmatrix} \begin{pmatrix} 1 \\ 1 \end{pmatrix} = \begin{pmatrix} 0 \\ 4 \\ 1 \\ 1 \end{pmatrix},$$

og vi kan nu opstille det første tableau på formen

b	A	0	0	0	0
$-c$	$2D$	$-A^t$	$A^t e$	I	$\triangle$

Med data som i problemet ovenfor får vi første tableau med udseendet:

7	1	2	1	0	0	0	0	0	0	0	0	0	0	0	0
4	−1	2	0	1	0	0	0	0	0	0	0	0	0	0	0
−3	−2	2	0	0	−1	1	0	1	0	0	0	−1	0	0	0
−4	2	−4	0	0	−2	−2	4	0	1	0	0	0	−1	0	0
0	0	0	0	0	−1	0	1	0	0	1	0	0	0	1	0
0	0	0	0	0	0	−1	1	0	0	0	1	0	0	0	1

Tableauet kan enten forståes umiddelbart, idet vi har opstillet vort ligningssystem i tableauform og så har tilføjet kunstige variable i det omfang, det var nødvendigt – der er to gode basisvariable i de første bibetingelsers slackvariable x_3 og x_4, men så kan vi ikke bruge μ_3 og μ_4, som ellers var fristende. Derfor nye variable u_3 og u_4. Vi kan heller ikke bruge μ_1 og μ_2, for der står jo $-$ i første søjle, så hele rækken skal ganges igennem med -1, hvorefter der må tilføjes kunstige slackvariable – eller alternativt, vi skal tilføje kunstige variable med koefficient -1 (som vist) og derpå skifte fortegn.

Det er iøvrigt netop dette tableau, der kommer frem, hvis man følger Wolfe-algoritmens generelle, men noget knudrede forskrift, således som angivet ovenfor. I vort eksempel, hvor vi har basis i slack-variable for de oprindelige bibetingelser, koger forskriften ned til at sætte fortegn på koefficienter til de nye kunstige slackvariable i overensstemmelse med fortegnene i $-c$.

Vi har altså startbasis i $x_3, x_4, u_1, u_2, u_3, u_4$, og vi vil gerne af med u'erne. Vi maximerer derfor funktionen $-\sum_j u_j$ og får det første tableau som:

Eksempel 4.2, fortsat

0	0	0	0	0	0	0	0	0	0	0	0	−1	−1	−1	−1
7	1	2	1	0	0	0	0	0	0	0	0	0	0	0	0
4	−1	2	0	1	0	0	0	0	0	0	0	0	0	0	0
3	2	−2	0	0	1	−1	0	−1	0	0	0	1	0	0	0
4	−2	4	0	0	2	2	−4	0	−1	0	0	0	1	0	0
0	0	0	0	0	−1	0	1	0	0	1	0	0	0	1	0
0	0	0	0	0	0	−1	1	0	0	0	1	0	0	0	1

Vi klargør tableauet, så der er 0 i øverste række over alle basisvariable (svarende til at lægge hver af de nederste fire rækker til den øverste:

7	0	2	0	0	2	0	−2	−1	−1	1	1	0	0	0	0
7	1	2	1	0	0	0	0	0	0	0	0	0	0	0	0
4	−1	2	0	1	0	0	0	0	0	0	0	0	0	0	0
3	2	−2	0	0	1	−1	0	−1	0	0	0	1	0	0	0
4	−2	**4**	0	0	2	2	−4	0	−1	0	0	0	1	0	0
0	0	0	0	0	−1	0	1	0	0	1	0	0	0	1	0
0	0	0	0	0	0	−1	1	0	0	0	1	0	0	0	1

Så kan vi gå i gang. Vi skal have 2. søjle ind (den er første søjle som har største koefficient 2, og der er ingen bindinger), og det betyder, at vi må pivotere om fjerde række. Derved får vi et nyt tableau, som ser således ud:

5	1	0	0	0	1	−1	0	−1	−0,5	1	1	0	−0,5	0	0
5	2	0	1	0	−1	−1	2	0	0,5	0	0	0	−0,5	0	0
2	0	0	0	1	−1	−1	2	0	0,5	0	0	0	−0,5	0	0
5	1	0	0	0	2	0	−2	−1	−0,5	0	0	1	0,5	0	0
1	−0,5	1	0	0	0,5	0,5	−1	0	−0,25	0	0	0	0,25	0	0
0	0	0	0	0	−1	0	1	0	0	1	0	0	0	1	0
0	0	0	0	0	0	−1	1	0	0	0	1	0	0	0	1

Her skal vi have x_1 ind i basis, og derved går x_3 ud. Der pivoteres, og i næste tableau (som ikke er vist) har λ_1 størst positiv koefficient (nemlig $1,5$), så den skal ind, og den fortrænger u_1. Efter pivotering ser tableauet således ud:

1	0	0	−0,2	0	0	−0,8	0,8	−0,4	−0,3	1	1	−0,6	−0,5	0	0
3	1	0	0,4	0	0	−0,4	0,4	−0,2	0,1	0	0	0,2	−0,5	0	0
3	0	0	−0,2	1	0	−0,8	0,8	−0,4	0,2	0	0	0,4	−0,5	0	0
1	0	0	−0,2	0	1	0,2	−1,2	−0,4	−0,3	0	0	0,4	0,5	0	0
2	0	1	0,3	0	0	0,2	−0,2	0,1	−0,05	0	0	−0,1	0,25	0	0
1	0	0	−0,2	0	0	0,2	−0,2	−0,4	−0,3	1	0	0,4	0,3	1	0
0	0	0	0	0	0	−1	1	0	0	0	1	0	0	0	1

Eksempel 4.2, fortsat

Her er det μ_3, som skal ind i basis, og det kan godt lade sig gøre, da x_3 ikke er i basis. Pivotelementet er i næstnederste række, så u_3 forlader basis, og da der allerede står en enhedsvektor i søjlen, sker der kun justeringer i øverste række, som bliver til:

0	0	0	0	0	0	-1	1	0	0	0	1	-1	-1	-1	0

Nu er det rækken svarende til λ_0, som skal ind i basis. Vi ser, at det mindste af tallene fremkommet ved at dividere elememt i b-søjlen med positivt element i søjlen er 0, som kommer fra nederste række. Det betyder, at vi skal pivotere i nederste række. Da vi allerede ser, at dette pivotstep fører til, at u_4 forlader basis, behøver vi ikke at gennemføre rækkeoperationerne, for vi ved nu, at vi har en løsning, og den kan jo allerede aflæses i b-søjlen (som ikke ændres pivotsteppet, check selv!).

Vi har altså en løsning med $x_1 = 3, x_2 = 2, x_4 = 3, \lambda_1 = 1,$ og $\mu_3 = 1$. Variablen λ_0 er i basis på nul-niveau, og de øvrige variable er naturligvis også 0.

3. Konveks simplex-metode

Hvis det foreliggende problem er af en type, hvor kriteriefunktionen ikke med rimelighed kan tilnærmes med enten en lineær eller en kvadratisk funktion, må man søge hjælp i andre metoder. Vi skal nøjes med en enkelt, nemlig den såkaldte konvekse simplex-metode introduceret af Zangwill i 1967.

Problemet, som skal løses, er følgende:

$$\begin{aligned} &\max f(x) \\ &\text{under bibetingelserne} \\ &Ax = b \\ &x \geq 0, \end{aligned}$$

hvor A er en $(m \times n)$-matrix og kriteriefunktionen f er C^1 (differentiabel med kontinuerte første afledede). Metoden er i sin tid udviklet for problemer med konveks kriteriefunktion, men den kan benyttes mere generelt.

Antag, at vi har en basisløsning x med tilhørende basis B. Vi skriver løsningen som (x_B, x_R), hvor x_B er vektoren svarende til de m variable i basis, og x_R er de resterende $n - m$ koordinater. Vi skriver matricen A tilsvarende som $A = [B\ R]$, hvorved bibetingelserne får formen

$$Ax = Bx_B + Rx_R = Bx_B + \sum_{j \in \mathcal{R}} x_j a_j = b,$$

hvor $\mathcal{R}$ er mængden af ikke-basis søjler i A, og a_j er den j'te søjle A. Da B er invertibel, har vi

$$x_B = \hat{b} - \sum_{j \in \mathcal{R}} x_j \hat{a}_j,$$

hvor $\hat{b}$ er værdierne af basisvariablene (venstre søjle i simplex-tableau'et), og $\hat{a}_j$ er j'te søjle im tableau'et. For den i'te basisvariabel giver det en værdi

$$x_{Bi} = \hat{b}_i - \sum_{j \in \mathcal{R}} x_j \hat{a}_{ij},$$

$i = 1, \ldots, m$.

I første omgang kan dette virke lidt omstændeligt, i og med at ikke-basis variablene er 0 i en basisløsning, men vor udledning skal læses en lidt anden vej: Den viser sammenhængen mellem x_B og x_R *også* hvis de sidste ændres en smule fra værdien 0. Dette vil vi udnytte ved at indsætte det sidste udtryk i kriteriefunktionen $f(x) = f(x_B, x_R)$, der med vor formel for x_B indsat bliver en funktion alene af x_R,

$$f(x_B, x_R) = \hat{f}(x_R).$$

Giver vi en ikke-basis variabel x_j en lille tilvækst, får vi en ændring i kriteriefunktionen på

$$\begin{aligned} \frac{\partial \hat{f}}{\partial x_j} &= \frac{\partial f}{\partial x_j} + \sum_{i=1}^{m} \frac{\partial f}{\partial x_{Bi}} \frac{\partial x_{Bi}}{\partial x_j} \\ &= \frac{\partial f}{\partial x_j} - \sum_{i=1}^{m} \frac{\partial f}{\partial x_{Bi}} \hat{a}_{ij}, \end{aligned}$$

eller, på matrixform,

$$\frac{\partial \hat{f}}{\partial x_j} = \frac{\partial f}{\partial x_j} - Df(x)_B \cdot \hat{a}_j.$$

Her har vi indført notationen $Df(x)_B$ for den del af gradientvektoren til f i x, som svarer til basisvariablene.

(Egentlig er udregningen ovenfor, der viser, hvad der kommer ud af det i form af øget kriterieværdi, hvis man øger en ikke-basisvariabel, ikke ny. Hvis vi lader $f(x)$ være den lineære funktion $c \cdot x$, får vi

$$\frac{\partial \hat{f}}{\partial x_j} = c_j - c_B \cdot \hat{a}_j,$$

som er koefficienterne til c-rækken i simplex-tableau'et.)

Vi kan nu finde en retning at bevæge os væk fra x i, nemlig i retningen af en ikke-basisvariabel x_j, for hvilken kriterieværdien vokser. Der er flere muligheder, så vi finder et index j_1 med

$$\frac{\partial \hat{f}}{\partial x_{j_1}} = \max_{j \in \mathcal{R}} \frac{\partial \hat{f}}{\partial x_j},$$

og et index j_2 med

$$\frac{\partial \hat{f}}{\partial x_{j_2}} x_{j_2} = \min_{j \in \mathcal{R}} \frac{\partial \hat{f}}{\partial x_j} x_j.$$

Den sidste udregning er med for det tilfælde, at det er optimalt at gøre x_j *mindre* (det er det ikke i allerførste trin, hvor de alle er 0, men i de senere trin kan vi have ikke-basisvariable med vægt forskellig fra 0). I dette tilfælde skal den partielle afledede være negativ.

Det ligger lige for, at hvis

$$\frac{\partial \hat{f}}{\partial x_{j_1}} > 0,\ \frac{\partial \hat{f}}{\partial x_{j_2}} x_{j_2} \geq 0,$$

da skal vi øge x_{j_1}. I det omvendte tilfælde skal vi formindske x_{j_2}. Hvis de begge har rigtigt fortegn (henholdsvis $>$ og $<$), er det lidt mere uklart, hvad man skal gøre; Zangwill foreslår at vælge den variabel, for hvilken det tilhørende udtryk er størst, og denne regel synes at have fungeret godt i praksis.

Den sidste mulighed, nemlig

$$\frac{\partial \hat{f}}{\partial x_{j_1}} \leq 0,\ \frac{\partial \hat{f}}{\partial x_{j_2}} x_{j_2} \geq 0$$

kan vises at være ensbetydende med, at Kuhn-Tucker betingelserne er opfyldt i punktet x. Hvis f er konkav, har man dermed en optimal løsning.

Dermed har vi fundet ud af, hvorledes vi bevæger os væk fra en løsning (medmindre den er optimal). Der resterer at finde ud af, *hvor langt* vi skal bevæge os. Det er nemlig ikke banalt som i simplex-metoden, hvor vi på grund af kriteriets linearitet altid skulle gå så langt, det var muligt, dvs. indtil en af basisvariablene blev trængt ud af basis. Det er ikke nødvendigvis nogen god ide generelt, for kriteriefunktionen kan vokse i starten, men derefter begynde at falde.

Der er tre tilfælde:

Tilfælde 1: x_j skal vokse, og der er mindst én koordinat af $\hat{a}_j$, som er positiv. I dette tilfælde vil en eller flere af basisvariablene aftage, når x_j vokser, og den første, som bliver 0, vil være x_{Bi_0} givet ved

$$\frac{x_{Bi_0}}{\hat{a}_{i_0 j}} = \min_{i:\hat{a}_{ij}>0} \frac{x_{Bi}}{\hat{a}_{ij}}.$$

Dette er velkendt fra simplex; vi kalder venstre side for $\bar{\lambda}$ og har, at x_j ikke kan øges med mere end dette, hvis løsningen skal forblive mulig.

Det er imidlertid som nævnt ikke sikkert, at man skal gå så langt; det afhænger af kriteriefunktionens værdi undervejs, og man beregner derfor $f(x + \lambda s)$, hvor s er vektoren (svarende til den retning, man bevæger sig) med

$$\begin{aligned}
&s_j = 1,\ s_{j'} = 0,\\
&j' \in \mathcal{R}, j' \neq j,\\
&s_{j''} = -\hat{a}_{ij},\ j'' \notin \mathcal{R},\ x_{j''} = x_{Bi},
\end{aligned}$$

og λ antager passende mange værdier i intervallet $[0, \bar{\lambda}]$. Det nye x bliver da bestemt ved det λ, for hvilken kriterieværdien er størst. Hvis maximum antages for $\lambda = \bar{\lambda}$, får vi dermed et basisskift. Ellers vil vi have den gamle basis, men nye ikke-basisvariable i x med vægt forskellig fra 0.

Tilfælde 2. x_j skal øges, men ingen koordinater af $\hat{a}_j$ er positive. Her gås der frem som før, bortset fra, at ligegyldig hvor stort et λ vi vælger, vil vi stadig have den gamle basis; vi skal derfor finde ny værdi af x ved at maximere $f(x + \lambda s)$ over alle $\lambda \geq 0$. Der vælges igen et passende antal værdier af λ; vi kan her risikere, at der slet ikke er noget maximum, og i så fald slutter algoritmen, idet er der ikke er løsning.

Tilfælde 3. x_j skal formindskes (hvilket naturligvis forudsætter, at den var positiv i forvejen). Nu kan x_j ikke mindskes lavere end til 0, men der kan endda være forhindringer for, at den kan gå så langt ned; find i_0 af

$$\frac{x_{Bi_0}}{\hat{a}_{i_0 j}} = \max_{i:\hat{a}_{ij}<0} \frac{x_{Bi}}{\hat{a}_{ij}};$$

kaldes venstresiden $\bar{\lambda}'$, og sætter $\bar{\lambda} = \min\{-\bar{\lambda}', x_j\}$, skal vi finde den maximale værdi af $f(x - \lambda s)$ for λ i intervallet $[0, \bar{\lambda}]$. Som i tilfælde 1 kan resultatet blive, at man stopper inden $\bar{\lambda}$ og bibeholder den gamle basis. Det vil man også gøre i nogle tilfælde, selvom man har maximum ved $\lambda = \bar{\lambda}$; for at x_j går ind i basis og driver en anden ud, kræves faktisk *både* at $\bar{\lambda}$ er optimal *og* at $-\bar{\lambda}' < x_j$. Under alle omstændigheder opdateres x, og vi har en ny mulig løsning, hvormed der kan fortsættes.

Eksempel 4.3

Betragt maximeringsproblemet

$$\max 10\log(x_1 + 1) - 2x_1 + x_1(x_2 + 1)^2$$

under bibetingelserne

$$x_1 + x_2 \leq 12$$
$$-x_1 + x_2 \leq 6$$
$$x_1, x_2 \geq 0$$

Vi ønsker at løse ved konveks simplex og opskriver straks det første tableau (hvor der er indført slack-variable):

	x_1	x_2	x_3	x_4
$x_{B1} = 12$	1	1	1	0
$x_{B2} = 6$	-1	1	0	1

Vi har her brugt notationen x_{Bi} for værdien af den $i$te basisvariabel; det er vigtigt at holde styr på disse. Det er også praktisk at have den pågældende variabel stående i over sin søjle.

Eksempel 4.3, fortsat

Vi har en basis i variablene x_3 og x_4 (således at første basisvariabel x_{B1} er x_3, anden basisvariabel x_4. Vi finder nu

$$\begin{aligned}\frac{\partial \hat{f}}{\partial x_1} &= \frac{\partial f}{\partial x_1} - Df(x)_B \cdot \hat{a}_1 \\ &= \frac{10}{x_1+1} - 2 + (x_2+1)^2 - (0,0)\cdot(1,-1) = 9\end{aligned}$$

og

$$\frac{\partial \hat{f}}{\partial x_2} = 2x_1(x_2+1) = 0.$$

Vi skal altså øge x_1. Vi finder grænserne (idet der er positive elementer i 1. søjle) til $12/1 = 12$, og vi skal altså løse

$$\max f((0,0,12,6) + \lambda(1,0,-1,1))$$

for $0 \leq \lambda \leq 12$. Bemærk, at de to sidste koordinater i vektoren $(1,0,-1,1)$ er første søjle med modsat fortegn, det er netop basiskoordinaternes afhængighed af x_1.

Når der indsættes, får vi

$$\max 10\log(\lambda+1) - \lambda$$

over $0 \leq \lambda \leq 12$. Vi differentierer og sætter lig nul, hvorved vi får

$$\frac{10}{\lambda+1} = 1$$

eller $\lambda = 9$, som er indenfor det tilladte λ-interval.

Vi opdaterer nu:

$$\begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 12 \\ 6 \end{pmatrix} + 9 \begin{pmatrix} 1 \\ 0 \\ -1 \\ 1 \end{pmatrix} = \begin{pmatrix} 9 \\ 0 \\ 3 \\ 15 \end{pmatrix}.$$

Da maximum af λ var inde i intervallet, kan vi beholde basis. Det nye tableau ser således ud:

	x_1	x_2	x_3	x_4
$x_{B1} = 3$	1	1	1	0
$x_{B2} = 15$	-1	1	0	1

Der er ikke sket noget inde i tableauet, for vi har ikke ændret basis. Men randsøjlen er ændret, fordi vi har opdateret vore basisvariable.

Vi går nu i gang med samme procedure som sidste gang:

$$\frac{\partial \hat{f}}{\partial x_1} = \frac{10}{x_1+1} - 2 + (x_2+1)^2 = 0$$

Eksempel 4.3, fortsat

(det er ikke overraskende, at vi får nul her, for sidste gang justerede vi netop på denne variabel),

$$\frac{\partial \hat{f}}{\partial x_2} = 2x_1(x_2+1) = 18.$$

Vi skal altså øge x_2. Grænserne findes som

$$\min\left\{\frac{3}{1}, \frac{15}{1}\right\} = 3,$$

og vi skal løse maximeringsproblemet

$$\max f((9,0,3,15) + \lambda(0,1,-1,-1))$$

over alle λ i intervallet $0 \leq \lambda \leq 3$. Ved indsætning af udtrykket for f får vi, at vi skal maximere $10\log 10 - 18 + 9(\lambda+1)^2$, og det ses direkte, at maximum fås ved at gå ud til højre endepunkt af intervallet, dvs. $\lambda = 3$. Ved opdatering af variablene fås

$$\begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{pmatrix} = \begin{pmatrix} 9 \\ 0 \\ 3 \\ 15 \end{pmatrix} + 3\begin{pmatrix} 0 \\ 1 \\ -1 \\ -1 \end{pmatrix} = \begin{pmatrix} 9 \\ 3 \\ 0 \\ 12 \end{pmatrix},$$

og skal også lave et basisskift: x_2 går ind i basis, x_3 går ud.

Det nye tableau ser således ud:

	x_1	x_2	x_3	x_4
$x_{B1} = x_2 = 3$	1	1	1	0
$x_{B2} = x_4 = 12$	-2	0	-1	1

Indmaden i tableauet er fundet ved sædvanligt pivotstep. Venstre søjle er den opdaterede basis.

Vi går i gang med et nyt beregningstrin:

$$\frac{\partial \hat{f}}{\partial x_1} = \frac{10}{x_1+1} - 2 + (x_2+1)^2 - [(2x_1(x_2+1),0)\cdot(1,-2)] = -57,$$

$$\frac{\partial \hat{f}}{\partial x_3} = 0 - [(2x_1(x_2+1),0)\cdot(1,-1)] = -72,$$

hvor første vektor i den kantede parentes har de to koordinater $\partial f/\partial x_2$ og $\partial f/\partial x_4$. Den anden af de to negative afledede er uinteressant, da x_3 er 0, men den første af dem tilsiger, at vi skal reducere x_1. For at finde grænserne for denne reduktion finder vi $12/-2 = -6$, så vi kan gå ned til $9-6=3$. Det betyder, at vi skal finde

$$\max f((9,3,0,12) - \lambda(1,-1,0,2))$$

på intervallet $0 \leq \lambda \leq 6$. Ved indsættelse af udtrykket for f bliver dette til

$$\max 10\log(10-\lambda) + 126 + 58\lambda + \lambda^2 - \lambda^3.$$

Eksempel 4.3, fortsat

Denne funktion lader sig ikke maximere med simple metoder, så vi opstiller en tabel over dens værdier i det tilladte interval:

λ	f
0	149,03
1	205,97
2	258,79
3	301,46
4	327,92
5	333,09
6	307,86

Vi finder λ mellem 4 og 6. Ved at checke den afledtes fortegn i punkterne eller ved yderligere opdeling kan man se, at maximum ligger mellem 4 og 5, omkring $4,5$ (faktisk er det $4,671$). Den nye løsning bliver da:

$$\begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{pmatrix} = \begin{pmatrix} 9 \\ 3 \\ 0 \\ 12 \end{pmatrix} - 4,5 \begin{pmatrix} 1 \\ -1 \\ 0 \\ 2 \end{pmatrix} = \begin{pmatrix} 4,5 \\ 7,5 \\ 0 \\ 3 \end{pmatrix};$$

basis er uændret. Det nye tableau bliver:

	x_1	x_2	x_3	x_4
$x_{B1} = x_2 = 7,5$	1	1	1	0
$x_{B2} = x_4 = 3$	-2	0	-1	1

Vi fortsætter med et nyt beregningstrin, og vi finder

$$\frac{\partial \hat{f}}{\partial x_1} \sim 0$$

(at det ikke passer helt, skyldes vor afrundingsfejl), samt

$$\frac{\partial \hat{f}}{\partial x_3} = 0 - [(2x_1(x_2+1), 0) \cdot (1, -1)] < 0,$$

og vi konkluderer, at løsningen er optimal.

4. Opgaver

1. En forbruger skal finde maximal nytte på en budgetmængde givet ved priser $p_1 = 2$ og $p_2 = 3$ på to fødevarer og indkomsten 5 samt en ekstra bibetingelser om at kalorieforbruget ikke må overstige 2000 kJ, idet varerne indeholder henholdsvis 900 og 1100 kJ pr. enhed. Forbrugerens nyttefunktion er af formen

$$U(x_1, x_2) = 3x_1 + 2x_2 + x_1x_2.$$

Find optimum ved kvadratisk programmering.

2. Løs problemet

$$\max \log(xy + 2) - y^2 + 2z$$

under bibetingelserne

$$x + y + z \leq 3$$
$$2x + z \leq 4$$
$$x, y, z \geq 0$$

med konveks simplex.

3. Løs følgende optimeringsproblem

$$\max xyz$$

under bibetingelserne

$$x + 2y + 3z \leq 10$$
$$2x + 3y + z \leq 10$$
$$x, y, z \geq 0$$

ved hjælp af konveks simplex.
(Det dur ikke at starte med $x = y = z = 0$, for så kan vores algoritme ikke komme videre. Overvej, hvad man kan gøre!)

5. Litteratur

Gennemgangen af kvadratisk programmering er inspireret af på fremstillingen i Simmons [1975], der også giver en alternativ algoritme til løsning af kvadratiske programmeringsproblemer.

For generelle ikke-lineære programmeringsproblemer findes flere forskellige algoritmer; konveks simplex er således blot en ud af mange, der alle er ganske komplicerede at anvende.

KAPITEL 5

Heltalsprogrammering

1. Optimering med heltalsrestriktioner

I det foregående har vi antaget, at de variable i vore optimeringsproblemer kunne antage alle reelle værdier, at der altså ikke var nogen indskrænkning i deleligheden. Det er der desværre ganske ofte i praksis. Visse variable kan f.eks. kun antage heltallige værdier, eller de har diskret variation, således at de antager én ud af endelig mange mulige værdier. I en sådan situation kan man ikke længere umiddelbart bruge metoderne fra de foregående kapitler. Der må særlige metoder til, og dem ser vi lidt nærmere på i dette kapitel.

Her er et par typiske eksempler på heltalsproblemer:

Eksempel 5.1

1. Et *0-1 knapsack problem* kan se ud som følger: Vi skal som det sig hør og bør på tracking i Nepal, og vor rygsæk skulle gerne indeholde følgende effekter:

	Vægt (kg)
Turistfører	0.5
Sovepose	1.2
Toiletsager	0.8
Walkman	0.7
Kinder mælkesnitter	0.5
Bærbar PC	1.7
Whisky	0.9
Noter til Operationsanalyse	1.4

Det oplyses fra arrangørerne, at rygsækkene ikke må veje mere end 3 kg, så der må vælges ud fra ovenstående effekter. Kriteriet for dette valg er, hvad det koster at købe tingene på stedet, og det oplyses at være følgende:

Eksempel 5.1,fortsat

	Pris ($)
Turistfører	25
Sovepose	35
Toiletsager	2
Walkman	40
Kinder mælkesnitter	25
Bærbar PC	620
Whisky	40
Noter til Operationsanalyse	100

Beslutningsproblemet ses nu at gå ud på at maximere værdien af den rygsæg der pakkes (således at udgifterne til at købe det, der ikke tages med, er så små som muligt). Hvis vi indfører en variabel x_i for effekt nr. i, således at $x_i = 1$ hvis vi tager i med og 0 hvis vi ikke tager i med, bliver det til

$$\begin{aligned}
&\max 25x_1 + 35x_2 + 2x_3 + 40x_4 + 25x_5 + 620x_6 + 40x_7 + 100x_8 \\
&\text{under bibetingelserne} \\
&0.5x_1 + 1.2x_2 + 0.8x_3 + 0.7x_4 + 0.5x_5 + 1.7x_6 + 0.9x_7 + 1.4x_8 \leq 3 \\
&x_i \in \{0,1\},\ i = 1,\ldots,8.
\end{aligned}$$

I den generelle udgave er der n forskellige projekter. Det j'te projekt har omkostninger a_j og værdi c_j. Hvert af projekterne kan enten iværksættes eller udelades; det er ikke muligt at gennemføre et projekt delvist. Der er et givet budget b til at betale for projekter med, og der ønskes iværksat projekter på en sådan måde, at den samlede værdi maximeres. Hvis vi indfører variable x_i, som tager værdien 0 eller 1, alt efter om projekt i gennemføres eller ej, får vi, at der skal vælges værdier af x_i, $i = 1,\ldots,n$, således at $\sum_{i=1}^{n} c_i x_i$ maximeres under bibetingelserne

$$\sum_{j=1}^{n} a_j x_j \leq b,\ x \in \{0,1\}^n.$$

2. Det følgende problem drejer sig om, hvordan man skal placere "faciliteter" som f.eks. brandstationer eller pølsevogne: Antag, at vi har en liste af mulige placeringer, eller formelt en mængde $N = \{1,\ldots,n\}$ af forskellige lokaliteter, hvor vi kan placere en af de pågældende faciliteter. Placeres den i lokaliteten j antages det at give anledning til en omkostning c_j (der afhænger af placeringen, det er f.eks. dyrere at leje lokaler på Rådhuspladsen end på Halmtorvet). Ved placering j er der en vis del af alle potentielle kunder (denne samlede kundemængde skrives som $M = \{1,\ldots,m\}$), formuleret som en given delmængde M_j af M, der betjenes fra j.

Problemet er nu at placere faciliteter så billigt som muligt, men samtidig således at alle kunder bliver betjent.

For at få dette til at blive et optimeringsproblem indføres variable x_j, indiceret ved de mulige lokaliteter i N, med værdier enten 0 eller 1, således at $x_j = 1$ hvis der placeres facilitet i lokalitet j, og $x_j = 0$ ellers. Betingelsen om, at alle kunder skal betjenes, er teknisk set at mængderne M_j for de j'er, der faktisk vælges, skal overdække M (således at hver $i \in M$ er i mindst ét M_j).

Eksempel 5.1,fortsat

Det lader sig mest bekvemt formulere ved brug af passende matricer: Vi lader A være ("incidens"-)matricen med elementer

$$a_{ij} = \begin{cases} 1 & \text{hvis } i \in M_j \\ 0 & \text{hvis } i \notin M_j \end{cases}$$

for $i = 1, \ldots, m, j = 1, \ldots, n$. Hvis $x = (x_1, \ldots, x_n)$ er en n-vektor af 0 eller 1, med 1-taller på alle de pladser i x, for hvilke den tilsvarende mængde M_j er med, så har vi en overdækning i den ovenfor beskrevne forstand netop hvis

$$Ax \geq 1$$

er opfyldt. Vi skal med andre ord løse

$$\begin{aligned} &\min\ c \cdot x \\ &\text{under bibetingelserne} \\ &Ax \geq 1 \\ &x \in \{0,1\}^n \end{aligned}$$

Hermed er problemet om valg af placering blevet omformuleret til et sædvanligt optimeringsproblem, dog med heltalsrestriktioner på de variable.

3. Betragt heltalsproblemet

$$\begin{aligned} &\max 0.18x_1 + x_2 \\ &\text{under bibetingelserne} \\ &x_1 \leq 5 \\ &x_2 \leq 3.75 \\ &0.25x_1 + x_2 \leq 4 \\ &x_1, x_2 \in \mathbb{Z}_+ \end{aligned}$$

Løses dette problem som et LP-problem uden hensyntagen til heltalsbetingelsen, fås løsningen

$$x_1 = 1,\ x_2 = 3.75,\ \text{kriterieværdi} = 3.9.$$

Når heltalsbetingelsen tilføjes, vil det være fristende simpelthen at runde ned til nærmeste heltal, således at vi får $x_1 = 1,\ x_2 = 3$.

Det er imidlertid *ikke løsningen* til heltalsproblemet: Ved at indsætte i bibetingelserne ser vi, at $x_1 = 4,\ x_2 = 3$ opfylder alle restriktionerne, og vi har, at den giver større værdi af objektfunktionen, nemlig 3.72.

Moralen er, at selvom LP-problemet er ret tæt på heltalsproblemet, behøver løsningen til LP-problemet bestemt ikke at ligge tæt på løsningen til heltalsproblemet. Dette er en af grundene til, at der må benyttes særlige metoder til at løse heltalsproblemer.

2. Metoder for heltalsprogrammering (1): Generel teori

De følgende afsnit er en ganske kort introduktion til teorien om løsning af heltalsproblemer. Der ses kun på problemer, hvor kriterie såvel som bibetingelser er givet ved lineære funktioner – naturligvis kombineret med heltalsrestriktioner.

Sådanne optimeringsproblemer kaldes forståeligt nok for ILP (Integer Linear Programming) problemer.

Egentlig er der ikke så meget teori, for der findes en lang række forskellige måder at tackle heltalligheden på, og teorien går nærmest ud på at systematisere dette virvar, vise at der, når det kommer til stykket, er et vist, ret beskedent udvalg af metoder, som kan bruges afhænging af problemets nærmere natur. Det endelige metodevalg er dog alligevel temmelig væsentligt for, hvor stor succes man vil have med sin problemløsning.

Den grundlæggende ide ved løsning af ILP-problemer er at anvende lineær programmering kombineret med passende argumenter til at håndtere heltalsrestriktionerne.

(a) Den nemmeste situation er den, hvor problemet

$$\begin{aligned} &\max c \cdot x \\ &\text{under bibetingelserne} \\ &Ax \leq b,\ x \in \mathbb{Z}_+^n \end{aligned} \tag{1}$$

har samme løsning som det tilhørende lineære programmeringsproblem, hvor x kan vælges fra $\mathbb{R}_+^n$, dvs. hvor man får noget heltalligt som løsning f.eks. ved simplexmetoden.

Man kan imidlertid ikke basere en generel forskrift på den slags held, så i det følgende antages det, at (1) ikke giver heltallig løsning.

(b) Man kan så forsøge at komme tæt på det oprindelige problem ved at løse *et andet* LP-problem, der måske er nemmere at gennemskue. Hvis man således tager en anden koefficientmatrix $\tilde{A}$ og en anden b-vektor $\tilde{b}$, valgt såled es, at

$$\tilde{A}x \leq \tilde{b}$$

opfyldes af alle x, der opfylder de rigtige bibetingelser, dvs. alle x fra mængden

$$S = \{x | Ax \leq b, x \in \mathbb{Z}_+^n\}.$$

I så fald vil en løsning til problemet

$$\max\{c \cdot x | \tilde{A}x \leq \tilde{b}, x \in \mathbb{R}_+^n\}$$

(hvis der findes en) give en kriterieværdi $\tilde{z}$, der er mindst lige så stor som den rigtige løsning (der er nemlig maximeret over en mængde af x'er, som indeholder S).

Det ser måske ikke ud af så meget, men det giver ihvertfald et check på, at der er løsninger i det oprindelige problem, samt en øvre grænse for kriterieværdien i optimum, såfrem det findes.

(c) Mere systematisk defineres en *relaxation* af optimeringsproblemet (1) som et optimeringsproblem

$$\max\left\{z_{RP}(x) | x \in S_{RP}\right\},$$

hvor $S_{RP} \subset \mathbb{R}^n_+$ er en mængde (af brugbare løsninger til det Relaxerede Problem), som opfylder

$$S \subset S_{RP}$$

(ligesom det var tilfældet ovenfor i (b)), og funktionen $z_{RP} : \mathbb{R}^n \to \mathbb{R}$ er en eller anden kriteriefunktion (til det Relaxerede Problem), der opfylder

$$c \cdot x \leq z_{RP}(x), \text{ alle } x \in S.$$

Formålet med relaxationen er det samme som i (b) ovenfor: Hvis problemet RP ikke har løsninger, da har (1) heller ingen løsninger. I modsat fald gælder

$$z^0 \leq z^0_{RP},$$

hvor z^0 og z^0_{RP} er maximumsværdierne af kriteriefunktionen i de to problemer.

I den generelle formulering behøver hverken kriteriefunktion eller bibetingelser i det relaxerede problem at være lineære; denne generelle formulering har dog mest interesse i den teoretiske diskussion (se næste afsnit), for i praksis vil man næppe være b egejstret for at bytte et lineært problem ud med et ikke-lineært. Det er mest naturligt at interessere sig for relaxationer i form af LP-problemer (simpelthen fordi LP-problemer er de nemmeste at have med at gøre).

(d) Den simpleste relaxation til (1) er problemet

$$\max\{c \cdot x | Ax \leq b\},$$

hvor man blot har droppet heltalsbetingelsen. Det er imidlertid ikke altid, at den er bedste bud, jvf. eksempel 3 ovenfor, og strengt taget bringer det os blot tilbage til punkt (a).

Næste skridt vil være at bibeholde kriteriefunktionen men ændre på bibetingelserne, og det kan der tit komme noget godt ud af, fordi mængden S af mulige x med heltalsbetingelserne opfyldt kan skrives på flere forskellige måder, nogle mere bekvemme end andre. En særlig simpel måde at omskrive på er at tilføje uligheder; denne standardmetode diskuteres i næste afsnit.

3. Metoder for heltalsprogrammering (2): Gyldige uligheder

En ulighed

$$\alpha_1 x_1 + \cdots + \alpha_n x_n \leq \beta$$

er *gyldig* (eng.: valid) for S (der blev indført i forrige afsnit som mængden af brugbare løsninger til heltalsproblemet (1)), hvis den er opfyldt for alle x fra S.

En gyldig ulighed kan tilføjes til et optimeringsproblem uden at dette ændres. Hvis man er særlig heldig, kan disse nye uligheder lede frem til heltalsløsninger

selvom de oprindelige ikke gjorde det, og det er begrundelsen for at tilføje nye bibetingelser (som man måske ellers ville synes at der var rigelig af fra starten).

Visse gyldige uligheder er naturligvis mere interessante end andre; oplagt nok er der ingen forskel på ulighederne

$$\alpha_1 x_1 + \cdots + \alpha_n x_n \leq \beta$$
$$k\alpha_1 x_1 + \cdots + k\alpha_n x_n \leq k\beta$$

for $k > 0$. En gyldig ulighed med koefficienter $(\alpha_1, \ldots, \alpha_n, \beta)$ dominerer en anden med koefficienter $(\alpha'_1, \ldots, \alpha'_n, \beta')$, hvis $\alpha_i \geq \alpha'_i$, alle i, og $\beta \leq \beta'$, med mindst én streng ulighed (den har altså større koefficenter på venstre side, mindre på højre). Hvis x tilfredsstiller den første af sådanne to uligheder, så må den nødvendigvis også tilfredsstille den anden, for

$$\sum_{i=1}^{n} \alpha'_i x_i \leq \sum_{i=1}^{n} \alpha_i x_i \leq \beta \leq \beta'$$

for alle $x = (x_1, \ldots, x_n) \in S$ (der er ikke-negativitetsbetingelser på alle variable). Det betyder, at den dominerede ulighed med koefficienter $(\alpha'_1, \ldots, \alpha'_n, \beta')$ ikke fortæller noget nyt i forhold til den første, eller omvendt at uligheden med koefficienter

$$(\alpha_1, \ldots, \alpha_n, \beta)$$

er mindst lige så god til at indsnævre mulighedsområdet.

En gyldig ulighed er *maximal* hvis den ikke er domineret af nogen gyldig ulighed. En omskrivning af et givet problem vil ofte gå ud på at lave maximale gyldige uligheder udfra de gyldige uligheder, som man har i forvejen. Der er flere forskellige måde at gøre dette på:

Afrunding. Her erstattes højresiden β i en heltallig ulighed med koefficienter

$$(\alpha_1, \ldots, \alpha_n, \beta)$$

med det største hele tal som er mindre eller lig β; dette tal skrives som $\lfloor \beta \rfloor$. Vi har at hvis

$$\alpha_1 x_1 + \cdots + \alpha_n x_n \leq \beta$$

er en gyldig ulighed med $\alpha_i \in \mathbb{Z}$ for each i, da er også

$$\alpha_1 x_1 + \cdots + \alpha_n x_n \leq \lfloor \beta \rfloor$$

en gyldig ulighed, idet der jo gælder, at for ethvert $x \in S$ er $\alpha \cdot x$ heltallig.

Linear kombinationer af gyldige uligheder: Hvis $(\alpha_1, \ldots, \alpha_n, \beta)$ og $(\alpha'_1, \ldots, \alpha'_n, \beta')$ er koefficientsæt for gyldige uligheder, da er enhver ikke-negativ linearkombination af disse,

$$(k\alpha_1 + k'\alpha'_1, \ldots, k\alpha_n + k'\alpha'_n, k\beta + k'\beta'),\ k, k', k_0, k'_0 \geq 0,$$

selv en gyldig ulighed. Man kan gå videre til såkaldte *disjunktive uligheder:* Hvis mulighedsområdet S kan skrives som en forening af to mængder S^1 og S^2, og

$$(\alpha_1^i, \ldots, \alpha_n^i, \beta^i),\ i = 1, 2,$$

er koefficienter for uligheder gyldige for henholdsvis $S^i, i = 1, 2$, hvor $S^1 \cup S^2 = S$, da er

$$\sum_{i=1}^{n} \min\{\alpha_i^1, \alpha_i^2\} x_i \leq \max\{\beta^1, \beta^2\}$$

en gyldig ulighed for S.

Superadditiv transformation. En funktion $F : \mathbb{R}^m \to \mathbb{R}$ med $F(0) = 0$ kaldes superadditiv, hvis

$$F(u_1 + u_2) \geq F(u_1) + F(u_2)$$

for alle $u_1, u_2 \in \mathbb{R}^m$. Det følger umiddelbart, at der for et positivt heltal K vil gælde $KF(u) \leq F(Ku)$.

Hvis F er superadditiv og ikke-aftagende, vil koefficientsættet $(\alpha_1, \ldots, \alpha_n, \beta)$ med

$$\alpha_j = F(a^j),\ j = 1, \ldots, n,\ \beta = F(b)$$

(her er a^j den $j$te søjle i matricen A, som definerer bibetingelserne i det oprindelige problem (1)) være en gyldig ulighed: Ethvert x i løsningsmængden er en vektor af hele tal, således at

$$\sum_{j=1}^{n} F(a^j) x_j = \sum_{j=1}^{n} [F(a^j) + \cdots + F(a^j)] \leq F(\sum_{j=1}^{n} a^j x_j) = F(b),$$

hvor summen i den kantede parentes består af x_j led.

Det ses således, at der er mindst tre forskellige måder at lave gyldige uligheder på. Der er naturligvis mange andre, men det kan vises, at disse tre metoder er udtømmende i den forstand, at de kan bruges til at få samtlige gyldige uligheder frem; for en given gyldig ulighed for

$$Ax \leq b,\ x \geq 0$$

med koefficienter $(\alpha_1, \ldots, \alpha_n, \beta)$ findes der en superadditiv funktion F så $\alpha_j \geq F(a^j)$, alle j, og $F(b) \leq \beta$. Man kan endda få uligheden frem alene ved anvendelse af linearkombination og afrunding, blot man gentager proceduren tilstrækkelig mange gange.

Vi skal ikke bruge dette resultat (for nærmere formulering og bevis henvises til f.eks. Nemhauser & Wolsey [1989]). Fordelen ved et resultat af denne type er, at når man i praksis leder efter gyldige uligheder, der er mere håndterlige end

de oprindelige, ved man, at et forholdsvis beskedent antal principper er nok til at bringe alle tænkelige uligheder frem.

4. Metoder for heltalsprogrammering (3): Dualitet

I stedet for at tilføje uligheder kan man overveje at ændre det givne optimeringsproblem ved at droppe passende mange af de givne bibetingelser. Det kan man selvfølgelig ikke gøre uden videre; ideen i det følgende er at lade dem indgå i objektfunktionen på samme måde som ved opstilling af Lagrange-funktion i sædvanlig analyse af nødvendige betingelser for optimering.

Antag, at det givne heltalsproblem kan skrives som

$$\begin{aligned}&\max c \cdot x\\ &\text{under bibetingelserne}\\ &A^1 x \leq b^1\\ &A^2 x \leq b^2\\ &x \in \mathbb{Z}_+^m\end{aligned}$$

hvor A^1 har m_1 rækker, A^2 har m^2 rækker. Det kan tænkes, at problemet, der fremkommer ved at udelade de første m_1 bibetingelser, er let at løse, hvorimod det samlede problem er vanskeligt (det kan undertiden ske, når det drejer sig om at maximere strømme i netværk (herom senere), og de sidste betingelser er balancebetingelserne i netværket, mens de første er yderligere restriktioner på strømmene.

Den generelle fremgangsmåde er at maximere Lagrange-funktionen

$$c \cdot x + \lambda \cdot (b^1 - A^1 x)$$

under bibetingelserne givet ved

$$A^2 x \leq b^2$$

over alle $x \in \mathbb{Z}_+^n$, for et $\lambda \in \mathbb{R}_+^m$. Det ses umiddelbart, at for ethvert $\lambda \geq 0$ vil en vektor x, som opfylder de oprindelige bibetingelser, også opfylde

$$c \cdot x + \lambda \cdot (b^1 - Ax^1) \geq c \cdot x,$$

således at den frembringer en relaxation af det oprindelige problem. Indtil videre er der dog ikke opnået meget ved omskrivningen, idet det der resterer at finde et egnet λ.

Generel dualitet. Eksemplet ovenfor fører naturligt ind på det mere generelle problem om at at løse et givet problem ved at udnytte information om det duale:

For et givet heltalsprogrammeringsproblem af typen

$$\begin{aligned}&\max c \cdot x\\&\text{under bibetingelserne}\\&x \in S\end{aligned}$$

kan der konstrueres *duale* problemer af typen

$$\begin{aligned}&\min z(u)\\&\text{under bibetingelserne}\\&u \in S^*\end{aligned}$$

således at $z(u) \geq c \cdot x$ for all $x \in S, u \in S^*$. Indtil videre er der ikke sagt noget om formen på restriktionsmængden S^*.

Problemer af denne art kaldes *svage* duale til det oprindelige problem. Betegnelsen dualitet er naturlig i lyset af, at det umiddelbart fra definitionen fås, at *hvis det duale problem har en mulig løsning u, da er $z(u)$ en øvre grænse for enhver mulig løsning af det oprindelige problem, og hvis det duale problem er ubegrænset (nedad), da har det oprindelige problem ingen mulig løsning*. Dette minder om, hvad man får ud af hovedsætningen for lineær programmering, men er naturligvis en del svagere, svarende til den mere generelle ramme her.

Løsningen til et dualt problem giver således øvre grænse for kriterieværdien. Kaldes den optimale værdi af kriteriefunktionen i henholdsvis det oprindelige heltalsproblem og det duale for z_{IP} og z_{DP}, fås det såkaldte *dualitetsgab*

$$\Delta = z_{DP} - z_{IP}.$$

Det er naturligvis mest spændende at se på et dualt problem, hvor dualitetsgabet er 0. Et sådant kaldes et *stærkt dualt* problem.

Som tidligere antages det, at bibetingelserne i det oprindelige problem giver mængden af brugbare løsninger

$$S = S(b) = \{x \in \mathbb{Z}_+^n | Ax \leq b\},$$

hvor vi har indføjet højresiden b i notationen for S for at understrege, at hvis b varierer, så vil også restriktionsmængden ændre sig. Vi definerer nu *værdifunktionen* ζ ved

$$\zeta(d) = \max\{c \cdot x | x \in S(d)\},$$

hvor $S(d) = \{x \in \mathbb{Z}_+^n | Ax \leq d\}$. Værdifunktionen angiver, hvor meget man får ud, målt ved kriteriefunktionen, af en given ressourcebegrænsning (formuleret ved d, højresiden i de lineære uligheder, der definerer de brugbare løsninger).

Vi vil antage det oprindelige problem normeret således at $\zeta(0) = 0$. Værdifunktionen ζ er iøvrigt superadditiv, for hvis $x_1 \in S(d_1)$ og $x_2 \in S(d_2)$, da er

$x_1 + x_2 \in S(d_1 + d_2)$, så summen af de to optimale kriterieværdier kan fås frem, og den er endda ikke nødvendigvis optimal.

Når man leder efter en øvre grænse for et heltalsprogrammeringsproblem, kan det gøres det ved at finde en funktion $g : \mathbb{R}^m \to \mathbb{R}$ således at

$$g(b) \geq \zeta(d) \text{ for alle } d \in \mathbb{R}^m$$

Med andre ord, vi har et dualt problem givet ved

$$\begin{aligned}
&\min g(d) \\
&\text{over alle funktioner } g \text{ med} \\
&g(d) \geq c \cdot x, \text{ alle } x \in S(d), d \in \mathbb{R}^m_+
\end{aligned}$$

Dette er et stærkt dualt problem, for man kan angive en funktion g (nemlig $g = \zeta$), hvor løsningen er lig med løsningen til det oprindelige problem. På den anden side er det uoverkommeligt at søge alle tænkelige funktioner igennem, så man må nøjes med en bestemt type af funktioner. Da værdifunktionen er ikke-aftagende, er det rimeligt at lade g være ikke-aftagende. Hvis g også kræves lineær, får vi det svage duale LP-problem, og det giver os ikke stærk dualitet. Kravet om linearitet er altså for restriktivt.

I lyset af den tidligere diskussion af gyldige uligheder ligger det i luften, at g skal være superadditiv. Det giver anledning til følgende *superadditive duale problem* (SDP):

$$\begin{aligned}
&\min F(b) \\
&\text{under bibetingelserne} \\
&F(a_j) \geq c_j \\
&F(0) = 0 \\
&F \text{ ikke-aftagende og superadditiv.}
\end{aligned}$$

Bemærk, at for en superadditiv funktion g, som løser dette problem, og for vilkårligt brugbart x gælder der

$$g(b) \geq g(Ax) \geq c \cdot x,$$

hvor vi først brugte at g var ikke-aftagende, og dernæst at den var superadditiv. Det viser, at det superadditive duale problem faktisk er et dualt problem. Det er endda stærkt dualt, for værdifunktionen ζ er i klassen, som opfylder bibetingelsern e. for $x \in \mathbb{Z}^n_+$ når $g(a_j) \geq c_j$ for hvert j (hvor a_j er søjlerne i A). Vi har nemlig

$$g(Ax) \geq \sum_{j \in N} g(a_j) x_j \geq \sum_{j \in N} c_j x_j = c \cdot x$$

på grund af superadditiviteten. På den anden side har vi, at $g(Ae_j) \geq c \cdot e_j$, hvor e_j er den $j$te enhedsvektor, er ensbetydende med $g(a_j) \geq c_j$ for alle j.

Vi har nu følgende version af dualitetssætningen fra før: *Hvis det oprindelige problem har en brugbar løsning, da har SDP en brugbar løsning, og der gælder for den optimale løsningsværdi w, at $w = F(b)$. Hvis det oprindelige problem ikke har brugbare løsninger, da er det duale ubegrænset nedadtil.*

Lagrange dualitet. Vi vender nu tilbage til problemet fra starten af afsnittet; antag at vi har lavet en Lagrange relaxation

$$\begin{aligned} &\max z(\lambda, x) = c \cdot x + \lambda(b^1 - A^1 x) \\ &\text{under bibetingelserne} \\ &A^2 x \leq b^2, x \in \mathbb{Z}_+^n; \end{aligned}$$

det resterende problem om at finde et λ, der minimerer den øvre grænse $\zeta_{LR}(\lambda)$ er så følgende:

$$\begin{aligned} &\min z_{LR}(\lambda) \\ &\text{over alle } \lambda \in \mathbb{R}_+^{m_1} \end{aligned}$$

hvor $z_{LR}(\lambda) = \max\{c \cdot x + \lambda(b^1 - A^1 x) | x \in \mathbb{Z}_+^n, A^2 x \leq b^2\}$. Dette minimeringsproblem kaldes det *Lagrange duale* til det oprindelige problem (med hensyn til bibetingelserne $A^1 x \leq b^1$).

Det Lagrange duale problem vil normalt være svagt dualt, så man kan ikke være sikker på at finde optimum af det oprindelige problem ved at løse det Lagrange duale. Der er dog i mange tilfælde en ret nem måde at finde en øvre grænse på; når man betragter funktionen $z(\lambda, x)$ for fastholdt x, er det en affin (dvs. lineær plus konstant) funktion af λ, og da Q er en endelig mængde, vil $z_{LR}(\lambda)$ være maximum af endelig mange affine funktioner og som sådan konvex (og stykkevis lineære). Minimering af sådanne funktioner under en betingelse om ikke-negativitet er et problem som i princippet er ret simpelt.

Man kan angive betingelser på det oprindelige problem, som sikrer, at det Lagrange duale er et stærkt dualt problem. Det skal vi dog ikke gå nærmere ind på.

5. Skæringsplan-algoritmer: Fractional cut

En intuitivt ret oplagt måde at løse et heltalsproblem på vil være at vælge en passende relaxation, løse det og checke, om løsningen er mulig i det oprindelige problem; er den ikke det, skal man så tilføje en gyldig ulighed som *ikke* er opfyldt at denne løsning, og fortsætte.

Dette er netop ideen i de såkaldte *skæringsplan-algoritmer,* blandt hvilke der her kun ses på en enkelt, som bruger såkaldte brøkdels-skæringsplaner (eng.: fractional cuts). Der tages udgangspunkt i en problemformulering af (næsten)

Eksempel 5.2

Vi ønsker at løse problemet

$$\begin{aligned}&\max x_1 + 3x_2 + 2x_3\\ &\text{u.bb.}\\ &x_1 + 7x_2 + 8x_3 \leq 7\\ &x_1 + x_2 + x_3 \leq 2\\ &x \in \{0,1\}^3\end{aligned}$$

ved brug af Lagrange-dualitet. Det Lagrange-duale problem (hvor den første af ulighederne er sat ind i kriteriefunktionen) er

$$\begin{aligned}&\min_{\lambda}\max_{x} x_1 + 3x_2 + 2x_3 + \lambda(7 - x_1 + 7x_2 + 8x_3)\\ &\text{u.bb.}\\ &x_1 + x_2 + x_3 \leq 2\\ &x \in \{0,1\}^3\end{aligned}$$

Dette problem er tilstrækkelig lille til at kunne håndteres direkte: Først laver vi en tabel over værdien af kriteriet for udvalgte λ-værdier:

	$\lambda = 1$	$\lambda = 0,5$	$\lambda = 0,1$	$\lambda = 0,01$
$x = (1,1,0)$	3	3,5	3,9	3,99
$x = (1,0,1)$	1	2	2,8	2,98
$x = (0,1,1)$	-3	1	**4,2**	**4,92**
$x = (1,0,0)$	**7**	**4**	1,6	1,06
$x = (0,1,0)$	3	3	3	3
$x = (1,0,0)$	1	1,5	1,9	1,9
$x = (0,0,0)$	**7**	3,5	0,7	0,7

I hver søjle er maximum (over x) skrevet med fede typer, og vi skal så vælge λ så at dette maximum bliver mindst muligt. Med forbehold for afrunding sker dette i $\lambda = 0,4$.

sædvanlig form:

$$\begin{aligned}&\max c \cdot x\\ &\text{under bibetingelserne}\\ &Ax = b,\ x \in \mathbb{Z}_+^m.\end{aligned}$$

Bemærk, at problemet er givet med lighedstegn i bibetingelserne. Det er velkendt fra de forrige kapitler, at det er nemt af få en sådan form frem, også hvis vi havde uligheder i starten, nemlig ved at tilføje slackvariable. Ved heltalsproblemer er der dog en detalje, som man skal være opmærksom på: Hvis problemet var givet ved uligheder, og der tilføjes slackvariable, skal man sikre sig, at alle løsninger til det oprindelige problem også implicerer heltallige løsninger i slackvariablene (for ellers har vi jo ikke formuleringen ovenfor). Det er ikke så svært at sikre sig det – man skal blot gange igennem så at *alle koefficienter i såvel A som b er heltallige*. For en sikkerheds skyld bør man altid checke denne heltallighed i startformuleringen, når man ønsker at bruge fractional cut, for ellers dur metoden ikke.

I første omgang lader man blot som ingenting med hensyn til heltalligheden og løser ved almindelig simplex. Antag at der er kørt simplex på problemet og man nået frem til en optimal basisløsning x^*. Ved eventuelt at flytte rundt på søjler i simplex-tableauet kan vi antage, at dette svarer til ligningssystemet

$$\{I|\hat{A}\}x = \hat{b},$$

hvor I er en enhedsmatrix svarende til basisvariablene, $\hat{A}$ er matricen svarende til ikke-basisvariablene, og $\hat{b}$ er vektoren af højresider i bibetingelserne.

Vi indfører nu *brøkdelen* af et reelt tal d som

$$[d] = d - \lfloor d \rfloor,$$

hvor $\lfloor d \rfloor$ som tidligere er det største heltal mindre eller lig med d. Vi har således $[17] = 0$, $[2.3] = 0.3$, men $[-2.3] = -2.3 - \lfloor -2.3 \rfloor = -2.3 - (-3) = 0.7$. Bemærk, at brøkdelen altid er ≥ 0.

Vælg nu en variabel x_i, som ikke er heltallig. Fra tableauet fås, at der gælder

$$x_i + \sum_{j \in J} \hat{a}_{ij} x_j = \hat{b}_i$$

for alle brugbare løsninger, hvor J som er alle ikke-basisvariable. Indsættes $\hat{a}_{ij} = [\hat{a}_{ij}] + \lfloor \hat{a}_{ij} \rfloor$ og tilsvarende $\hat{b}_i = [\hat{b}_i] + \lfloor \hat{b}_i \rfloor$, bliver denne ligning til

$$x_i + \sum_{j \in J} \lfloor \hat{a}_{ij} \rfloor x_j + \sum_{j \in J} [\hat{a}_{ij}] x_j = [\hat{b}_i] + \lfloor \hat{b}_i \rfloor.$$

Da der for alle i og j trivielt gælder $\lfloor \hat{a}_{ij} \rfloor \leq \hat{a}_{ij}$, fås trivielt, at

$$x_i + \sum_{j \in J} \lfloor \hat{a}_{ij} \rfloor x_j \leq \hat{b}_i;$$

nu bruger vi, at enhver løsning er heltallig; hvis venstre side er $\leq \hat{b}_i$, er den også $\leq \lfloor \hat{b}_i \rfloor$. Dermed har vi

$$x_i + \sum_{j \in J} \lfloor \hat{a}_{ij} \rfloor x_j \leq \lfloor \hat{b}_i \rfloor;$$

Tilbage i uligheden ovenfor bliver så

$$\sum_{j \in J} [\hat{a}_{ij}] x_j \geq [\hat{b}_i],$$

som er en gyldig ulighed for vort heltalsproblem.

Et skæringsplan (eller et *snit*) af denne type kaldes et *(Gomory) brøkdels-snit*. Ved at bruge denne type snit kan det oprindelige heltalsproblem løses som en endelig følge af LP-problemer. Vi har altså her en metode med hel generel anvendelighed. Desværre har det vist sig, at metoden ikke virker alt for godt i praksis; det er ofte en fordel at benytte specielle former for skæringsplaner, hvor man udnytter særlige egenskaber ved det givne problem.

Eksempel 5.3

Betragt følgende ILP-problem:

$$\begin{aligned}
&\max 2x_1 + x_2 \\
&\text{under bibetingelserne} \\
&3x_1 + 4x_2 \leq 10 \\
&x_1 + x_2 \leq 15 \\
&x_1, x_2 \in \mathbb{Z}_+
\end{aligned}$$

Vi vil løse dette ved hjælp af fractional-cut metoden. Vi starter med at opstille simplex-tableauet:

	2	1	0	0
10	3	4	1	0
15	1	1	0	1

Her er alle koefficienter heltallige, som de skal være (ellers måtte vi gange i rækkerne med passende tal) . Vi går nu igang med helt sædvanlig simplex, hvilket her vil sige, at vi skal have første variabel ind i basis. Det nye tableau bliver (der er her og i det følgende afrundet efter anden decimal):

$-6,67$	0	$-1,67$	$-0,67$	0
$3,33$	1	$1,33$	$0,33$	0
$11,67$	0	$-0,33$	$-0,33$	1

Dermed har vi nået maximum, men løsningen er ikke heltallig. Vi indfører derfor en ny bibetingelse: vi tager udgangspunkt i sidste række i tableauet og skriver kun brøkdelene ind (idet brøkdelen af f.eks. $11,67$ er $0,67$, og brøkdelen af $-0,33$ er $0,67$ – vi går *ned* til nærmeste heltal, i dette tilfælde -1, og tager derpå forskellen op til det tal, vi startede med). I den nye søjle skal vi have koefficienten -1 svarende til at den nye bibetingelse er givet ved et $\geq$. Ialt fås nyt tableau:

Eksempel 5.3,fortsat

$-6,67$	0	$-1,67$	$-0,67$	0	0
$3,33$	1	$1,33$	$0,33$	0	0
$11,67$	0	$-0,33$	$-0,33$	1	0
$0,67$	0	$0,67$	$0,67$	0	-1

Som tableauet står her, er vi i første omgang kørt en smule fast, for vi har nemlig ikke nogen basis (svarende til en enhedsmatrix i passende søjler). Det kan vi dog hurtigt klare ved at *skifte fortegn i den nye række:*

$-6,67$	0	$-1,67$	$-0,67$	0	0
$3,33$	1	$1,33$	$0,33$	0	0
$11,67$	0	$-0,33$	$-0,33$	1	0
$-0,67$	0	$-0,67$	$-0,67$	0	1

Nu har vi en basis, men til gengæld står der noget negativt i b-søjlen, og det må der ikke, når vi kører simplex. Derfor går vi over til *dual simplex*. Betingelserne er iorden. Vi har jo, at optimalitetsbetingelserne er opfyldt (alle koefficienter i c-rækken er ikke-positive). Vi skal altså skifte den negative række, svarende til den nye slack-variabel x_5, ud af basis, og hvis vi søjlevis dividerer elementer i c-rækken med negative elementer i 3. række, får vi minimum i 3. søjle, som bliver vort pivot-element. Det nye tableau bliver:

-6	0	-1	0	0	-1
3	1	1	0	0	$0,5$
12	0	0	0	1	$-0,5$
1	0	1	1	0	$-1,49$

Her er løsningen både mulig og optimal. Den er endda heltallig, så dermed har vi fået en løsning til vort oprindelige problem.

Hvis løsningen ikke havde været heltallig, kunne der fortsættes med et nyt cut. Man kan i almindelighed ikke regne med at nå den eksakte optimale løsning, men må stoppe, når man efter at have gennemført en række cut ser, at man nærmer sig en bestemt heltallig løsning.

6. Branch-and-bound

Den sidste type af algoritmer for heltalsproblemer, som vil blive behandlet, benytter

en meget almindelig fremgangsmåde til at reducere vanskelige beregningsproblemer. Hvis et problem er "stort" i den forstand, at det er svært at håndtere, kan man forsøge sig med at dele det givne problem op i mindre, som hver især er lettere at klare.

Umiddelbart kan det se ud, som om der er vundet meget lidt ved dette, for godt nok for vi to problemer af rundt regnet halv størrelse (hvad der her skal forstås ved "størrelse", er uklart, men vi vender tilbage til det; indtil videre må man klare sig med intuitionen), men til gengæld er der to problemer i stedet for tidligere ét. Pointen er, at vanskelighederne ved at løse problemer som regel vokser mere end lineært med størrelsen; det er derfor lettere at løse to små end et, der er dobbelt så stort. Det er et generelt princip om "del-og-hersk" ved konstruktion af algoritmer.

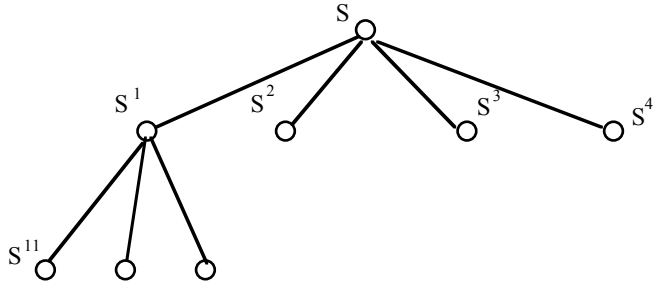
Princippet i *branch-and-bound* metoden er, at den oprindeligt givne mængde S af mulige løsninger til heltalsproblemet skal kunne skrives som en foreningsmængde af passende delmængder S^i for $i = 1, \ldots, k$. Vi har da, at

$$\max_{x \in S} c \cdot x = \max_{i=1,\ldots,k} \max_{x \in S^i} c \cdot x,$$

dvs. at finde den optimale løsning svarer til at finde optimal løsning på hver af mængderne $S^1, \ldots, S^k$, og dernæst tage den største.

Fordelen ved denne fremgangsmåde er – foruden den ovenfor nævnte om besværet ved beregningerne – at man ofte kan undgå at beregne visse af delproblemerne, nemlig når man har en øvre grænse for delproblemets maximum, som afslører at dette problem ihvertfald ikke kan give den generelt bedste løsning.

I anvendelser af metoden skal opdelingen i delproblemer ofte gentages, svarende til at mængden S opdeles stadig finere. Denne successive opdeling kan illustreres som vist i figur 1, hvor vi har den givne mængde S øverst, og i hvert efterfølgende lag har inddeling af en mængde i laget ovenfor. Figuren er en særlig type graf, et såkaldt *træ*. Punktet svarende til S kaldes *roden,* og forbindelseslinierne mellem punkter (der svarer til at nederste punkt er en delmængde af øverste) kaldes *hjørner.*



Figur 1

Under løsningen af det oprindelige problem startes der med at inddele fra roden og til det næstøverste lag. Derefter ses der på hvert punkt i dette lag og man går

videre med inddeling. Hvis det ved behandlingen af et punkt kan fastslås, at der ikke er behov for videre inddeling af en mængde, siger vi, at træet kan *beskæres* i det pågældende punkt. Der gælder, at træet kan beskæres i punktet svarende til en mængde S^i hvis

– der ikke er nogen mulige løsninger i S^i (dvs. $S^i = \emptyset$),

– en optimal løsning til problem $\max_{x \in S^i} c \cdot x$ kendes,

– det vides, at $\max_{x \in S^i} c \cdot x \leq \max_{x \in S} c \cdot x$.

I hvert af disse tilfælde kan man stoppe videre undersøgelser af træet i den gren, som indeholder punktet, og koncentrere sig om resten af træet.

Det kan ofte lade sig gøre at afgøre, om et træ kan beskæres i et punkt, uden at man behøver løse de tilknyttede heltalsproblemer til bunds. Hvis problemet er lineært, kan man benytte den lineære relaxation til at give en øvre begrænsning af de optimale værdier; denne øvre grænse vil så kunne justeres nedad efterhånden som man arbejder sig igennem træet. Omvendt vil man kunne få en nedre grænse ved at vælge en vilkårlig mulig løsning i hvert delproblem og tage objektfunktionens værdi på denne; optimum må nødvendigvis være større eller lig den største af de værdier, man får på denne måde.

Ved at forsætte beskæring og opdeling kan man dermed i princippet få nedre og øvre grænse for optimum til at mødes, og man har da en optimal løsning. I praksis er det ikke helt ligegyldigt, *hvordan* man foretager opdelingen; hvis vi har at gøre med variable, der kun antager værdierne 0 og 1, kan man lade første inddeling være

$$S^0 = \{x \in S | x_1 = 0\},\ S^1 = \{x \in S | x_1 = 1\},$$

anden inddeling

$$S^{00} \ldots, S^{11},\ S^{ij} = \{x \in S^i | x_2 = j\},\ i, j = 0, 1,$$

og så fremdeles. Her vil valget af rækkefølge af variablene godt kunne spille en rolle, og generelt, når variablene kan antage andre heltalsværdier, vil det være af stor betydning, at der er valgt en god måde at dele op i mindre problemer på. Dette svarer til den generelle indsigt fra de andre metoder: De generelle metoder er ofte afhængige af, at der vælges en god fremgangsmåde ved deres anvendelse; der er ikke for heltalsproblemerne, som tilfældet er for almindelige LP problemer, en enkelt generelt god metode til at løse disse problemer.

Eksempel 6.2

Betragt følgende eksempel på et heltalsproblem:

$$\max 4x_1 + 4x_2 + 3x_3 + x_4$$

under bibetingelserne

$$\begin{aligned} 3x_1 - 5x_2 + 7x_3 + 4x_4 &\leq 10 \\ x_2 + 3x_3 - 2x_4 &\leq 3 \\ 2x_1 + 2x_2 + 2x_4 &\leq 2 \\ x &\in \{0, 1\}^4 \end{aligned}$$

Eksempel 6.2, fortsat

Vi vil løse dette problem ved branch-and-bound metoden; det er oplagt at definere træet ved at lade S^0 og S^1 være delmængderne af S med $x_1 = 0$ og $x_1 = 1$, derefter at opdele efter om x_2 er 0 eller 1 osv.

1. trin: Vi starter med at sætte $x_1 = 0$. Det oprindelige problem bliver da til:

$$\max 4x_2 + 3x_3 + x_4$$

under bibetingelserne

$$\begin{aligned} -5x_2 + 7x_3 + 4x_4 &\leq 10 \\ x_2 + 3x_3 - 2x_4 &\leq 3 \\ 2x_2 + 2x_4 &\leq 2 \\ x &\in \{0,1\}^3 \end{aligned}$$

Ved at bruge simplex på det tilhørende LP-problem får vi, at dette problem har en løsning med

$$x_2 = 1,\ x_3 = 0.67,\ x_4 = 0,\ \text{kriterieværdi} = 6.$$

Prøver vi dernæst tilfældet $x_1 = 1$, får vi ved indsættelse i det oprindelige problem, at vi skal løse:

$$\max 4x_2 + 3x_3 + x_4$$

under bibetingelserne

$$\begin{aligned} -5x_2 + 7x_3 + 4x_4 &\leq 7 \\ x_2 + 3x_3 - 2x_4 &\leq 3 \\ 2x_2 + 2x_4 &\leq 0 \\ x &\in \{0,1\}^3 \end{aligned}$$

Løsningen til det tilhørende LP-problem fås ved simplex-metoden til at være:

$$x_2 = 0,\ x_3 = 1,\ x_4 = 0,\ \text{kriterieværdi} = 3.$$

Til de 3 fra denne løsning skal nu lægges de 4, som vi får fra den oprindelige kriteriefunktion for indsat $x_1 = 1$, ialt 7. Konklusionen på dette er, at vi har en heltallig (og dermed mulig) løsning til det oprindelige problem (nemlig den sidst fundne), som er bedre end LP-løsningen til varianten for $x_1 = 0$. Med andre ord, *træet kan beskæres* for $x_1 = 0$, som dermed er udelukket i det følgende.

2. trin: Vi prøver nu med næste variabel x_2: For $x_1 = 1, x_2 = 0$ får vi problemet

$$\max 3x_3 + x_4$$

under bibetingelserne

$$\begin{aligned} 7x_3 + 4x_4 &\leq 7 \\ 3x_3 - 2x_4 &\leq 3 \\ 2x_4 &\leq 0 \\ x &\in \{0,1\}^2 \end{aligned}$$

med løsningen

$$x_1 = 1,\ x_2 = 0,\ x_3 = 1,\ x_4 = 0,\ \text{kriterieværdi} = 3.$$

Dette er iøvrigt løsningen fra 1.trin.

Eksempel 6.2, fortsat

For $x_1 = 1, x_2 = 1$ fås problemet

$$\begin{aligned} &\max 3x_3 + x_4 \\ &\text{under bibetingelserne} \\ &\begin{aligned} 7x_3 + 4x_4 &\leq 12 \\ 3x_3 - 2x_4 &\leq 4 \\ 2x_4 &\leq -2 \end{aligned} \\ &x \in \{0,1\}^2 \end{aligned}$$

Her er der ingen ikke-negativ løsning til sidste bibetingelse, så den tilhørende mængde S^{11} er tom. Vi kan derfor beskære ved S^{11}.

3.trin: Vi prøver nu at sætte x_3 til henholdsvis 0 og 1:

For $x_1 = 1$, $x_2 = 0$ og $x_3 = 0$ får vi problemet

$$\begin{aligned} &\max x_4 \\ &\text{under bibetingelserne} \\ &\begin{aligned} 4x_4 &\leq 7 \\ -2x_4 &\leq 2 \\ 2x_4 &\leq 0 \end{aligned} \\ &x \in \{0,1\}^2 \end{aligned}$$

Vi ser umiddelbart, at $x_4 = 0$. Den tilhørende kriterieværdi er

$$4 + 0 + 0 + 0 = 4;$$

Vi har allerede tidligere haft en heltalsløsning med kriterieværdi 7, så den foreliggende kan kasseres; træet kan beskæres i S^{100}.

For $x_3 = 1$ får vi igen et problem, hvor sidste ulighed kun har løsningen $x_4 = 0$. Den tilhørende løsning $(x_1, x_2, x_3, x_4) = (1, 0, 1, 0)$ har, som vi tidligere så, værdien 7. Da dette er den eneste mulighed tilbage, er det løsningen.

7. Opgaver

1. En kommune skal indkøbe kunst til at hænge på væggene i rådhuset. Der er afsat 110.000 kr. til dette formål, og man forhandler med tre forskellige kunstnere. Hver af dem har enhedspris på deres værker, nemlig henholdsvis 20, 15 og 12 (tusind kroner).

Den første kunstners billeder har format $0,4 \times 0,4$, den andens har størrelsen $2,3 \times 2$, og den tredies har størrelsen $2,1 \times 3$ (alle mål i m). Man har udregnet, at der af æstetiske hensyn skal være 10 gange billedets størrelse i tomt vægareal omkring billedet; der er ialt 322m^2, som skal udsmykkes.

Der er en begrænsning i det samlede indkøb på 9 billeder; videre er det en betingelse, at hver af de tre kunstnere vil være repræsenteret i det endelige billedvalg.

Find det optimale valg, når det fra kommunalbestyrelsen er meddelt, at man finder den anden kunstners billeder dobbelt så gode som de to andres.

2. Som et led i et økologisk eksperiment støttet af WHO og NATO er det besluttet at oprette et selvforsynende samfund på Vejrø i Kattegat (den, som man sejler forbi med Ask og Urd, når de altså sejler).

Husdyrproduktionen er baseret på nyeste videnskabelige resultater: Der kan opfodres enten kalkuner (x_1), får (x_2) eller grise (x_3); man har ved grundige undersøgelser fundet ud af, at næringsværdien for den samlede øbefolkning er $\ln(x_1 + 1)$ enheder ved produktion af grise, $x_2 + \sqrt{x_2}$ ved produktion af får, og $2 + x_3 - e^{-2x_3}$ ved produktion af grise.

Husdyrene skal dele staldfaciliteter på ialt $27m^2$, og man har fundet frem til, at en kalkun optager $0,7m^2$, et får $1,1m^2$, og en gris $0,9m^2$. Affaldsstofferne fra husdyrholdet skal genanvendes, og anlægget har en begrænsning på 2.300 kg. Der må påregnes et affald på 70 kg for en kalkun, 95 kg for et får og 165 kg for et svin.

På grund af den særlige status som forsøg skal der som minimum holdes 2 stk. af hvert af de nævnte dyr.

Find optimum (man er tilfreds med langsigtsgennemsnit, så der kræves ikke heltallige løsninger).

3. Et mindre sommerland overvejer at åbne en lille zoologisk have. Der kan anskaffes en eller to af følgende dyrearter, idet der for hver af dem er opgivet ugentlige omkostninger, pladsbehov og afføring:

	Omkostninger (1000 kr.)	Plads m^2	Afføring (kg.)
Elefant	2,6	51	63
Næsehorn	3,4	25	42
Søelefant	9,2	10	12
Giraf	4,3	8	14
Pingvin	4,3	8	0,4
Zebra	2,2	18	10
Krokodille	8,8	22	25
Flodhest	2,1	20	30
Egern	0,1	2	0,1
Chimpanse	3,3	20	10

Det kræves, at der er mindst 6 forskellige dyrearter, og der er kun 108 m^2 til rådighed. Af hensyn til det omgivende land må der ikke udledes mere end 86 kg. dyreafføring om ugen. Hvorledes skal den zoologiske have sammensættes, når der ønskes minimale omkostninger?

4. Et nystartet kondicenter planlægger sit indkøb af maskiner. Der er mulighed for at købe tre slags apparater (1,2 og 3), der koster henholdsvis 300.000, 200.000 og 100.000 kr. Apparaternes strømforbrug er 1.2 kW for apparat 1, 1.3 kW for apparat 2 og 1.7 kW for apparat 3.

Centrets bankforbindelse har oplyst, at der ikke kan påregnes kredit højere end 1.5 mio.kr.; videre er den installerede strømforsyning af en sådan art, at der ikke kan trækkes mere end 8 kwh.

Det er nødvendigt for centrets profilering udadtil, at det råder over alle tre typer af apparater. Desuden har centrets ledelse oplyst, at den anser type 1 og 3 for nogenlunde lige gode til at trække kunder til, mens 3 maskiner af type 1 svarer til 2 af type 2.

Find det optimale indkøb.

8. Litteratur

Der findes adskillige fremragende lærebøger om heltalsprogrammering, som er et område, hvor der forskes intensivt, således f.eks. Nemhauser og Wolsey (1988), Papadimitriou og Steiglitz (1982) og Schrijver (1986). De fleste af disse dækker bredere end blot de emner, der er omtalt i kapitlet.

KAPITEL 6

Travelling Salesman

1. Det klasssiske TSP-problem

Et nærmest klassisk eksempel på et heltalsproblem er det såkaldte TSP (Travelling Sales Person) problem. Her er der givet en mængde af lokaliteter og et antal veje mellem disse lokaliteter. Til hver sådan vej er der knyttet en omkostning (som kan tænkes at udtrykke rejsetid eller rejseomkostninger), og problemet går ud på at finde en tur, startende ved by 1, som passerer samtlige byer netop en gang og returnerer til by 1, valgt således at man får minimal samlet rejsetid.

For at formulere sagen som et heltalsproblem, indfører vi først noget notation: Vi betegner mængden af lokaliteter med V og mængden af veje med E; vi antager, at der kun er én vej fra lokalitet i til lokalitet j (ellers vælger vi den billigste ud), så hver $e \in E$ kan skrives som (i, j), hvor i er startlokalitet, j slutlokalitet. Der indføres variable

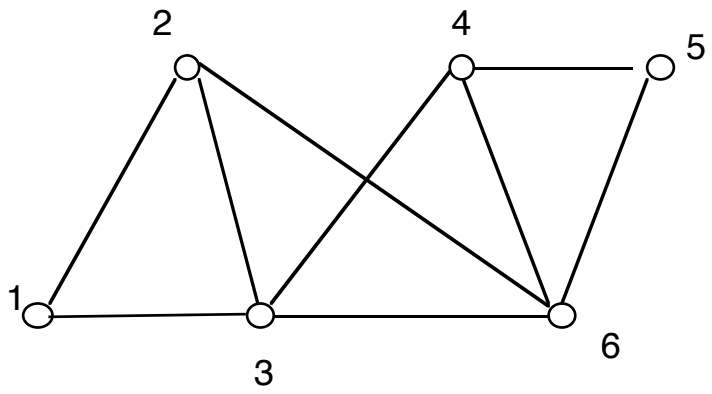
$$x = x_{(i,j)},\ (i,j) \in V,$$

til beskrivelse af en tur, hvor $x_{ij} = 1$, såfremt j følger umiddelbart efter i på turen, og $x_{ij} = 0$ ellers. Hvis x er en rundtur, må den altså opfylde

$$\sum_{i:(i,j)\in E} x_{ij} = 1 \text{ for all } j \in V$$
$$\sum_{j:(i,j)\in E} x_{ij} = 1 \text{ for all } i \in V$$

der siger, dels at ethvert $j \in V$ besøges netop én gang, dels at ethvert $i \in V$ forlades i en eller anden retning (turen går ikke i stå). Hertil kommer heltalsbetingelsen $x_{ij} \in \{0, 1\}$ for alle (i, j).

Der mangler dog stadig noget i at formulere problemet færdigt; ligningerne ovenfor vil være opfyldt f.eks. hvis der findes særskilte mindre ture, der tilsammen kommer alle byer igennem, således som turene $1 \to 2 \to 3 \to 1$ og $4 \to 5 \to 6 \to 4$ i figur 1; sådan en samling mindre ture er imidlertid ikke en rundtur i den forstand, vi ønsker (der kræves flere adskilte TSP'er til at betjene dem). Den foreslåede (fedt optrukne) "tur" giver ikke mulighed for at komme fra byerne $\{1, 2, 3\}$ til byerne $\{4, 5, 6\}$, og derfor kan den ikke bruges. Generelt gælder der for enhver delmængde



Figur 1

U af byerne i V med $2 \leq |U| \leq |V| - 2$ (hvor $|A|$ betegner antallet af elementer i mængden A), at mindst en vej fra U til $V \setminus U$ må være i brug, dvs.

$$\sum_{(i,j) \in E, i \in U, j \notin U} x_{ij} \geq 1.$$

Alt i alt kan vi da formulere travelling salesman problemet som

$$\min \sum_{(i,j) \in \mathcal{E}} c_{ij} x_{ij}$$

over alle x som opfylder bibetingelserne ovenfor. Bemærk, at der er temmelig mange bibetingelser, nemlig dels de to første og dels en for hver delmængde af V med mellem 2 og $|V| - 2$ elementer. Ialt er der altså nær ved $2^{|V|}$ bibetingelser, hvilket er temmelig mange og specielt har den uheldige egenskab, at antallet vokser meget kraftigt med antallet af byer. Der er altså en vis forhåndsformodning om, at traveling salesman problemet kan være ganske besværligt at håndtere, noget vi senere hen skal vende tilbage til.

Eksempel 7.1

Vi vil i det følgende vise, hvorledes man kan gå frem ved løsning af et TSP problem af den type, som blev introduceret i starten af kapitlet. Metoden er *branch-and-bound* som forklaret i det foregående kapitel. Vi benytter et konkret eksempel som ramme for diskussionen:

Et rejsebureau planlægger en rundrejse i Danmark for en japansk turist. Rejsetiden skal være så lille som mulig, men turisten skal se København, Odense, Århus, Skagen og Legoland. Udgangspunktet er Tabel 1 over rejsetider.

Man skal ikke hæfte sig for meget ved tabellens tal, ligesom det måske i eksemplet ikke er så oplagt, hvorfor det er umuligt at rejse direkte fra København til Legoland. Eksemplet viser blot, hvorledes et problem, som oprindeligt var givet ved en graf (rejsemål med rejseruter imellem, rejsetider angivet ved hver rejserute) på naturlig måde kan præsenteres ved den udvidede incidensmatrix. Der står ikke tal i diagonalen; det svarer til, at der ikke i den underliggende graf er løkker – og hvis der er, skal de ihvertfald ikke bruges i rejseruten, for de fører ikke nogen steder hen.

Eksempel 7.1, fortsat

Tabel 1. Rejsetider

Til: / Fra:	Kbhvn.	Odense	Århus	Skagen	Legol.
Kbhvn.	–	1	4	2	umulig
Odense	1	–	3	1	3
Århus	5	4	–	4	2
Skagen	2	1	5	–	1
Legoland	3	2	3	1	–

For at løse problemet om den hurtigste rejserute vil det i det konkrete tilfælde være overkommeligt at checke samtlige mulige rundture. Det vil dog hurtigt (dvs. når antallet af rejsemål vokser) blive et kolossalt omfattende arbejde. Det er derfor vigtigt at have en systematik, der kan bruges generelt.

Trin 1. Reducér hver række og søjle med det mindste element. Hvis der findes en tur (hvilket i vor matrix-fremstilling vil sige fem elementer, som aldrig står i samme række eller søjle), som har 0 i alle elementer, er den optimal.

I eksemplet sætter vi ∞ i diagonalen og ved de umulige rejseruter, og vi trækker dernæst mindste element fra i hver række; det giver os tabel 2.

Tabel 2. Rejsetider reduceret med mindste element i hver række.

Til: / Fra:	K	O	Å	S	L	Reduktion
K	∞	0	3	1	∞	1
O	0	∞	2	0	2	1
Å	3	2	∞	2	0	2
S	1	0	4	∞	0	1
L	2	1	2	0	∞	1

Af tabellen kan vi se, at enhver rejserute, som udgår fra København, må kræve en rejsetid på mindst 1; tilsvarende for Odense, Skagen og Legoland, mens alle rejseruter ud af Århus kræver rejsetid 2. Ialt har vi dermed, at man ikke kan komme under 6 timer i samlet rejsetid.

Vi gennemfører nu samme procedure med søjlerne og får tabel 3.

Den nederste række angiver den mindste rejsetid, der kræves, når man forlader et rejsemål; dette giver yderligere 2 timer til vor nedre begrænsning på samlet rejsetid, der altså mindst må være 8 timer.

Eksempel 7.1, fortsat

Tabel 3. Rejsetider reduceret både i rækker og søjler

Til: / Fra:	Kbhvn.	Odense	Århus	Skagen	Legol.
Kbhvn.	∞	0	1	1	∞
Odense	0	∞	0	0	2
Århus	3	2	∞	2	0
Skagen	1	0	2	∞	0
Legoland	2	1	0	0	∞
Reduktion	0	0	2	0	0

Vi skal nu udvælge en bestemt rejserute mellem to mål, og derefter tilpasse resten. Det er naturligt at vælge denne rute, så at den hører blandt dem, der giver kort samlet rejsetid:

Trin 2. Find omkostningen ved ikke at bruge en rute med reduceret omkostning 0 som minimum af omkostningerne i den række plus omkostningerne i den søjle, der svarer til det pågældende element.

Tabel 4. Omkostninger ved ikke-brug for ruter med reduceret omkostning 0

Til: / Fra:	Kbhvn.	Odense	Århus	Skagen	Legol.
Kbhvn.	∞	1 0	1	1	∞
Odense	1 0	∞	0 0	0 0	2
Århus	3	2	∞	2	2 0
Skagen	1	0 0	2	∞	0 0
Legoland	2	1	0 0	0 0	∞

Trin 3. Vælg et element med højeste omkostninger for ikke-brug og skriv en ny tabel op, hvor denne rute bliver brugt.

Eksempel 7.1, fortsat

I tabel 4 vælger vi ruten fra Århus til Legoland, idet omkostningerne ved ikke at bruge denne rute er maximal. Vi kan nu opstille en ny tabel, hvor Århus-rækken og Legoland-søjlen ikke er med. Her skal vi sætte omkostningerne for rejseruten Legoland-Århus til ∞, for den kan vi jo ikke bruge, når vi allerede har valgt Århus-Legoland. Resultatet bliver tabel 5.

Tabel 5. Rejserutetabel givet at der vælges ruten Århus – Legoland.

Til: / Fra:	Kbhvn.	Odense	Århus	Skagen
Kbhvn.	∞	0	1	1
Odense	0	∞	0	0
Skagen	1	0	2	∞
Legoland	2	1	∞	0

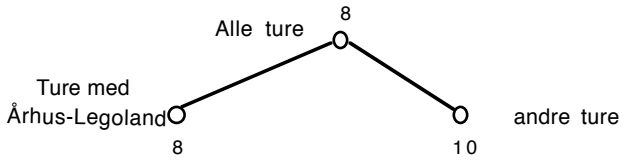
Tabel 6. Rejserutetabel givet Århus – Legoland, omkostninger ved ikke-brug

Til: / Fra:	Kbhvn.	Odense	Århus	Skagen
Kbhvn.	∞	1 0	1	1
Odense	1 0	∞	1 0	0 0
Skagen	1	1 0	2	∞
Legoland	2	1	∞	1 0

Trin 4. Vi har nu en ny tabel svarende til den vi startede med (bortset fra at der mangler en række og en søjle). Vi skal her igen klargøre, således at der er et nul i hver række og hver søjle (trække mindste element fra i hver række/søjle), men det er allerede i orden i eksemplet. I vort problem betyder dette, at 8 stadig er en nedre grænse for omkostningen ved ture, der bruger ruten Århus-Legoland.

Eksempel 7.1, fortsat

Før vi går videre, kan vi opsummere det hidtidige arbejde som følger: Vi startede med at skulle undersøge alle ture; på nuværende tidspunkt har vi heraf udskilt to grene, nemlig en svarende til alle de ture, der bruger ruten Århus-Legoland, og en anden svarende til alle de andre:



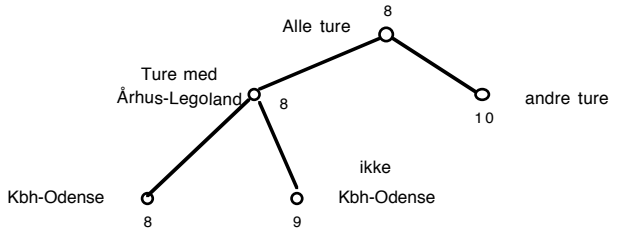
Den mindst mulige omkostning ved den første gren er 8, ved den anden 10.

Trin 5. Som tidligere vælger vi nu en rute med mindst mulig omkostning og analyserer omkostninger i den tabel, der fremkommer, når række og søjle svarende til dette valg fjernes. I eksemplet vælger vi ruten København-Odense, se tabel 6. Resultatet bliver den nye tabel 7 med tre rækker og søjler.

Tabel 7. Rejserutetabel givet at der vælges ruter
Århus – Legoland, København – Odense

Til: / Fra:	Kbhvn.	Århus	Skagen
Odense	∞	0	0
Skagen	1	2	∞
Legoland	2	∞	0

Da omkostningerne ved ikke at vælge denne rute er 1, får vi følgende nye udgave af vort træ (med minimale omkostninger føjet på):



Denne tabel kan reduceres (2. række har ingen nuller), hvilket giver os tabel 8 med en reduktion på 1. Det betyder, at den mindst mulige omkostning ved ture indeholdende ruterne Århus-Legoland og København-Odense er 9.

I tabel 8 kan vi vælge 3 nuller således at alle rækker og søjler er repræsenteret – svarende til en tur med minimale omkostninger; det giver os ruterne Skagen-København, Odense-Århus og Legoland-Skagen. Ialt har vi dermed fundet en optimal tur:

København-Odense-Århus-Legoland-Skagen-København

med en samlet omkostning på 9. Vi kan se, at denne tur er optimal, fordi enhver tur svarende til en anden gren i træet vil koste mindst 9, og vi har fundet den billigste i denne gren.

Eksempel 7.1, fortsat

Tabel 8. Reduceret tabel givet ruterne
Århus – Legoland, København – Odense

Til: / Fra:	Kbhvn.	Århus	Skagen	Reduktion
Odense	∞	0	0	0
Skagen	0	1	∞	1
Legoland	2	∞	0	0

Det er ikke udelukket, at der kan være andre optimale løsninger. Vi ved, at de nødvendigvis må indeholde ruten Århus-Legoland, for ellers er omkostningerne jo mindst 10. Men vi kan ikke være sikre på, at de indeholder København-Odense. Det kan man checke ved at opskrive den tilsvarende tabel for situationen

Århus-Legoland, ikke København-Odense

(tabel 9), hvor der er kommet et ∞ ind på pladsen svarende til København-Odense. Der kan nu køres videre på sædvanlig facon (reduktion, valg af nulelement med størst omkostning ved ikke-brug); man kan overbevise sig om, at der faktisk er en alternativ tur med samme minimale omkostning 9.

Tabel 9. Rejserutetabel givet at der vælges
Århus – Legoland, *ikke* København – Odense

Til: / Fra:	Kbhvn.	Odense	Århus	Skagen
Kbhvn.	∞	∞	1	1
Odense	0	∞	0	0
Skagen	1	0	2	∞
Legoland	2	1	∞	0

Det fornemmes, at denne algoritme er en omstændelig sag. Det hænger blandt andet sammen med, at branch-and-bound algoritmer undertiden vil skulle vende tilbage til tidligere undersøgte situationer og køre i ny retning. Dermed er der behov for at holde styr på et ganske stort talmateriale. Vi skal senere se, at Travelling Salesman faktisk i en ret præcis forstand kan vises at være et "svært" (NP-komplet) problem.

2. Vehicle Routing

Problemet fra det foregående afsnit har, som man måske kan fornemme, givet inspiration til ganske mange nyskabelser indenfor diskret optimering, og som teoretisk problemstilling indtager det derfor en plads for sig selv. Til gengæld kan man sige, at der er forholdsvis få direkte praktiske anvendelser; i de fleste tilfælde er der særlige forhold til stede, som gør, at det ikke er et helt rendyrket TSP problem, man har at gøre med.

Der kan således være mere end en sælger til at besøge alle byerne, så at problemet handler om at finde ruter til hver af dem, der tilsammen opfylder kravet om at alle byer skal besøges, og det skal foregå billigst muligt. Men yderligere kan der være krav om, at besøget af de enkelte byer skal foregå i bestemte tidsintervaller, "time windows", så at man ikke blot skal besøge byerne, men også gøre det på bestemte tidspunkter. Derved bliver problemstillingen noget mere kompliceret, men til gengæld får vi en bedre model af den virkelighed, som vi gerne vil kunne lave om på.

Problemer af denne art kaldes for *vehicle routing problems* (VRP) og har været studeret siden midten af 60'erne. Vi har her et antal kunder, indiceret $i = 2, \ldots, n$, et antal køretøjer $k = 1, \ldots, m$, samt et depot, som har index $i = 1$. Hver af kunderne har en given efterspørgsel q_i, og der er kørselsomkostninger c_{ij} ved at køre fra i til j, og hver enkelt køretøj har en kapacitet Q_k, $k = 1, \ldots, m$. Man plejer at nummerere kunderne så $q_2 > q_3 > \cdots > q_n$. Det basale VRP problem drejer sig nu om at tilordne ruter til køretøjerne således at alle kunder får deres efterspørgsel opfyldt (varerne findes, det er ikke det, som er problemet), så at intet køretøj belastes mere end svarende til kapaciteten, og så at den samlede kørselsplan bliver billigst muligt.

Dette basale VRP problem kan formaliseres på flere forskellige måder. Her er en forholdsvis direkte måde at skrive det op på: Vi indfører variable x_{ijk}, som er givet ved

$$x_{ijk} = \begin{cases} 1 & \text{hvis køretøj } k \text{ besøger kunde } j \text{ umiddelbart efter kunde } i, \\ 0 & \text{ellers,} \end{cases}$$

og tilsvarende lader vi y_{ik} være defineret som

$$y_{ik} = \begin{cases} 1 & \text{hvis kunde } i \text{ besøges af køretøj } k \\ 0 & \text{ellers.} \end{cases}$$

Vi kan nu formulere VRP som

$$\min \sum_{i,j} c_{ij} \sum_{k} x_{ijk}$$

under bibetingelserne

$$\sum_k y_{ik} = \begin{cases} m & \text{for } i = 1 \\ 1 & \text{for } i = 1, \ldots, m \end{cases}$$

$$\sum_i q_i y_{ik} \leq Q_k,\ k = 1, \ldots, m,$$

$$\sum_i x_{ijk} = \sum_j x_{ijk} = y_{ik},\ i = 1, \ldots, n,\ k = 1, \ldots, m,$$

$$\sum_{i,j \in S} x_{ijk} \leq |S| - 1 \text{ for alle } S \subset \{2, \ldots, n\},\ k = 1, \ldots, m,$$

hvor det er underforstået, at y_{ik} og x_{ijk} er variable som tager værdier i $\{0, 1\}$

Det første sæt bibetingelser sikrer, at hver kunde er sat på ruteplanen for et eller andet køretøj (depotet skal besøges af dem alle sammen), og det næste er kapacitetsbegrænsningerne for køretøjerne. Herefter kommer bibetingelser, som sikrer, at ethvert køretøj k, som besøger en kunde i, også forlader kunden igen ("flow conservation"), og til sidst har vi en bibetingelse af den type, som vi stødte på i forbindelse med TSP, nemlig udelukkelse af ture, der ikke er ægte i den forstand, at der køres i ring udenfor depotet ("subtour elimination").

Det er faktisk den sidste type bibetingelse, der gør problemet vanskeligt, idet man jo i princippet skal checke alle delmængder af kunder, hvad der selv for et beskedent antal kunder bliver en meget krævende opgave. I det følgende skitseres, hvorledes en algoritme til løsning af VRP kan skrues sammen; vi skal ikke gå i detaljer, for beregningsarbejdet bliver under alle omstændigheder drabeligt, og VRP er et område, hvor der satses på *heuristikker,* algoritmer, der ikke kan vises at løse det konkrete problem, men som kommer tæt på. Det vender vi tilbage til.

Algoritmer for VRP: Hvis man ser nærmere på formuleringen af VRP-problemet ovenfor, kan man se, at det i realiteten består af to forskellige – men naturligvis indbyrdes afhængige – problemer: Der er for det første et tilordnings- eller *assignmentproblem* som går ud på at tilordne køretøjer til kunder, svarende til de første sæt to bibetingelser, og dernæst et TSP problem for hvert køretøj med tilordnede kunder. Det betyder, at vi kan omskrive problemet til

$$\max \sum_k f_k(y_k)$$

hvor $y_k = (y_{1k}, \ldots, y_{nk})$, og hvor $f_k(y_k)$ defineres som den minimale omkostning (altså optimalværdien) af et TSP problem med byerne $\{i \mid y_{ik} = 1\}$ foruden depotet. Med andre ord, f_k er defineret ved

$$f_k(y_k) = \max \sum_{i,j} c_{ij} x_{ijk}$$

under bibetingelserne

$$\sum_i x_{ijk} = \sum_j x_{ijk} = y_{ik},\ i = 1, \ldots, n,$$

$$\sum_{i,j \in S} x_{ijk} \leq |S| - 1 \text{ for alle } S \subset \{2, \ldots, n\},$$

for $k = 1, \ldots, n$.

Omskrivningen giver imidlertid ikke noget simpelt svar på, hvordan man finder løsningen, for kriteriefunktionerne f_k, som giver optimalværdien af et TSP som funktion af dets parametre, har en kompliceret form. Men man kan benytte sig af duale variable – ved hver løsning af TSP-underproblemerne for passende y_k, $k = 1, \ldots, n$, kan man få duale variable (som angiver værdien af, at bibetingelserne ændres en smule, her værdien af at ændre på de enkelte y_{ik} (fortolket: besparelsen for køretøj k ved at blive af med en given kunde), og disse værdier kan man så bruge til at bytte ud mellem de enkelte delproblemer (hvad der svarer til at bytte kunder mellem de enkelte køretøjer), så at den samlede omkostning minimeres.

Det lyder for så vidt intuitivt, men der er nogle komplikationer undervejs, ikke mindst i forbindelse med, at TSP-underproblemerne jo er heltalsproblemer, så der kommer ikke automatisk veldefinerede duale variable ud af løsningen. Det kan man så få ved at gå bort fra heltalsrestriktionen, men så har man på den anden side tilføje noget andet, så at løsningen alligevel bliver heltallig, og her bruger man ofte lineære uligheder af passende form (i terminologien fra forrige kapitel bliver det til en skæringsplans-algoritme). Dermed er den simple intuitive struktur i algoritmen jo gledet noget i baggrunden, og beregningsarbejdet bliver ganske stort.

Heuristiske metoder. I betragtning af, at en præcis beregning af løsningen til et VRP ihvertfald vil komme til at indeholde et TSP – og endda formodentlig som en subrutine, der skal gennemføres adskillige gange, er det fristende at lede efter metoder, der giver en acceptabel tilnærmelse til løsningen, uden at det helt præcist er en løsningsmetode. Det er ikke en spørgsmål om bekvemmelighed; VRP problemer af passende størrelse vil slet ikke kunne løses med realistiske begrænsninger med hensyn til tid og maskinkapacitet

I det følgende skitseres nogle af de heuristikker, der har været foreslået til løsning af VRP.

Besparelses-algoritmen. Denne metode (der skyldes Clark & Wright, 1964) er blandt de først udviklede og mest benyttede. Fremgangsmåden er som følger: Der startes med et system af ruter $1 \to i \to 1$, så at hver kunde forsynes særskilt; denne ruteplan er næppe mulig (så skulle der være lige så mange køretøjer som kunder), men det er også blot udgangspunktet.

Trin 1: Her udregnes alle *besparelser* $s_{ij} = c_{1i} - c_{ij} + c_{j1}$ for par (i, j) af kunder. Besparelsen s_{ij} svarer til gevinsten ved at knytte forbindelsen $i \to j$, således at man laver ruten $1 \to i \to j \to 1$ i stedet for at forsyne i og j særskilt i to ruter $1 \to i \to 1$ and $1 \to j \to 1$.

Trin 2: Alle besparelser ordnes i aftagende rækkefølge.

Trin 3: Begyndende fra oven i denne rækkefølge, gør følgende:

Trin 4: Find den første brugbare forbindelse $i \rightarrow j$, som kan indføjes i den hidtil konstruerede rute lige før køretøjet vender tilbage til depotet.

Trin 5: Hvis der ikke er en sådan forbindelse, vælg første brugbare forbindelse $i \rightarrow j$, som kan bruges til at starte en ny rute.

Trin 6. Fortsæt trin 4 og 5 så længe der kan vælges nye forbindelser.

Undervejs i algoritmen bør man checke de ture, der er konstrueret, for at sikre sig, at de foreliggende køretøjer kan gennemføre dem. Da algoritmen starter med en plan, som ikke er mulig, har man ikke nogen sikkerhed for at ende med en plan, hvor der er køretøjer nok – der kan tænkes at restere nogle enkelt-kunde-ruter, som der ikke er køretøjer til. Løsningen efterlader dermed nogle kunder uden betjening, hvad der betyder, at bibetingelsen $\sum_k y_{ik} = m$ kan være overtrådt for visse i.

Sweep algoritmen (Gillett og Miller, 1974). Denne algoritme konstruerer ruter ved at lade en lyskegle feje rundt fra depotet: Vi antager at kunderne er placeret i planen med depotet i nulpunktet, og vi kan da angive kundernes beliggenhed ved såkaldte polære koordinater (r_i, θ_i), hvor r_i er afstanden til i (fra depotet) og θ_i er den retning, hvori man skal bevæge sig fra depotet for at komme til i, hvor retningen måles som vinklen fra retningen til en given valgt referencekunde i^*. Vi antager, at kunderne er nummereret sådan at $\theta_2 \leq \ldots \leq \theta_n$.

Der gås nu frem som følger:

Fase 1:

Trin 1. Vælg et ubrugt køretøj k.

Trin 2. Begynd ved den første kunde i, som endnu ikke er i ruteplanen (hvilket vil sige den kunde udenfor planen, som har lavest værdi af θ_i. Lav derefter en kundekreds $\{i, i+1, i+2, \ldots\}$, idet der fortsættes til kapaciteten af køretøjet k er udtømt.

Trin 3. Hvis alle kunder eller køretøjer er brugt, gås der til fase 2, ellers gås der tilbage til Trin 1.

Fase 2:

Trin 4. Løs TSP for kunderne "fejet sammen" til hvert køretøj i Fase 1.

Der kan laves variationer af sweeping-algoritmen ved at ændre referencekunden i^* (der jo har rollen som udgangspunkt for første kundekreds) såvel som måden at ordne kunderne på.

CMT 2-fase metoden (efter Christofides, Mingozzi og Toth, 1979); sweeping-algoritmen er iøvrigt også en 2-fase metode, hvor der først findees kundekredse og derefter løses TSP).

Fase 1:

Trin 1. Vælg en endnu ikke placeret kunde i og et køretøj k til at betjene ruten med.

Trin 2. Vælg en ubrugt kunde j som knyttes til i's kundekreds bestemt ved at j har lavest værdi af en entreomkostningsvariabel beregnet i forhold til i,

indtil kapacitetsgrænsen for k er nået. Hvis alle kunder er placeret, eller alle køretøjer fyldt, gås til næste trin, ellers gentages fra trin 1.

Trin 3. For hver kundekreds frigøres alle andre kunder end den initiale, og for hver fri kunde undersøges det, om hun kan indgå i en anden. Vælg den billigste (m.h.t. entreomkostninger).

Trin 4. Allokér den kunde, der i forrige runde havde billigste entreomkostninger.

Trin 5. Trin 3 og 4 for hver kunde, hvis tidligere bedste entre ikke længere er mulig, og indtil der ikke er flere måder at lade kunder indgå på.

Fase 2:

Trin 6. Løs TSP for hver af de fundne kundekredse.

Metoden går, som den forrige, ud på først at finde kundekredse, og derefter tilpasse ruten indefor hver kundekreds. Til forskel fra den foregående laves kundekredsene ikke én gang for alle, men der er en efterfølgende runde, hvor det undersøges, om kundekredsene kan forbedres. At der er tale om heuristik snarere end præcis løsning, ses af at de ret arbitrært fundne nøglepersoner i hver kundekreds bibeholdes – det gøres for at sikre, at der er en veldefineret omkostning forbundet med at tilordne en kunde til en køretøj i en situation, hvor ret mange af de øvrige kunder hos køretøjet er usikre i den forstand, er deres placering ikke er optimeret.

En senere heuristik (Fisher og Jaikumar's 2-fase metode) erstatter den indledende tilpasning af lundekredse med en anvendelse af en kendt algoritme, såkaldt generaliseret assignment (se kapitel 11).

3. Opgaver

1. En virksomhed producerer plastvarer og har en kontrakt om leverancer af 6 forskellige varetyper nemlig sæbeskåle (S), grydeskeer (G), askebægre (A), kopper (K), tallerkener (T) og fade (F). Alt produceres på samme maskine, og mellem produktion af hver varetype går der tid (angivet nedenfor i minutter) til rengøring, omstilling af støbeforme osv.

	S	G	A	K	T	F
S	–	10	8	6	12	20
G	8	–	12	10	7	10
A	15	7	–	15	3	8
K	7	9	9	–	9	13
T	9	5	5	11	–	12
F	10	12	10	12	12	–

Find optimal rækkefølge i produktionen af de 6 varetyper.

2. Ved fastlæggelsen af kontrolproceduren før start af et trafikfly har det vist sig, at der er sammenhæng mellem de målinger, der netop er foretaget, og sandsynligheden

for at overse en fejlfunktion ved umiddelbart efterfølgende måling. Man har fundet frem til sandsynlighederne for denne fejltype, der er som vist i tabellen (der er fem funktioner, der testes, og tabellen angiver sandsynligheden (i promille) for at overse fejl i funktionerne angivet i søjlerne, når man netop har fuldendt inspektionen af funktionerne angivet i rækkerne):

10	8	20	30	15
5	10	12	20	8
7	13	10	21	14
15	14	13	9	28
12	14	12	8	18

Fra de tekniske specifikationer ved man, at funktion 1-3 er 10 gange så sikker som de to sidste funktioner.

Find den checkprocedure, der maximerer sikkerheden.

4. Litteratur

TSP problemet er behandlet i de fleste fremstillinger af matematisk programmering. Der findes også en (særdeles anbefalelsesværdig) bog om TSP, nemlig Lawler, Lenstra, Rinnooy Kan og Shmoys (1985).

KAPITEL 7

Dynamisk programmering

1. Indledning

I de foregående eksempler har vi set på metoder til løsning af optimeringsproblemer, der hver især kunne bruges på en hel klasse af problemer, karakteriseret (især) ved kriteriefunktionernes form: Hvis kriteriet var lineært (og bibetingelserne ligeledes), havde vi at gøre med LP, hvis det var en kvadratisk eller måske en konveks funktion, var der tale om kvadratisk eller konveks programmering. Kriteriefunktionen var afgørende for, hvilken metodik vi ville benytte os af.

I dette kapitel drejes problemet en smule rundt; vi vil ikke længere fiksere på kriteriefunktionens egenskaber, men til gengæld vil vi interessere os noget mere for forekomsten af andre slags struktur i problemstillingen. Sagt noget mindre kryptisk: I adskillige problemer er der en helt naturlig opdeling af det generelle optimeringsproblem i mindre problemer relateret til hvert sit tidspunkt. En systematisk udnyttelse af denne tidsopdeling – teknisk formuleret, af problemets rekursive struktur – danner baggrund for de metoder, der går under betegnelsen dynamisk programmering.

At der er en naturlig tidsmæssig struktur i mange praktiske problemer, er næppe noget, der kræver meget argumentation. Produktions- og lagerproblemer involverer variable dateret på hvert sit tidspunkt over en lang fremtidig periode. Beslutningsproblemet på hvert givet tidspunkt kan naturligvis ikke løses uden at det ses i sammenhæng med tidligere og senere beslutninger. Der er altså normalt ikke mulighed for banalt at spalte problemet op i mindre. Men ved en behændig udnyttelse af strukturen kan man alligevel ofte nå en løsning på en måde, der ret meget ligner en sådan opsplitning. Der er ikke noget generel metode på samme måde som ved LP-problemer, hvor vi har simplex-metoden til rådighed; der er ganske vist noget generel formalisme, som vi skal støde på senere, men hovedreglen er, at der må udvises en smule omhu i problemets behandling, for at man kan høste den fulde fordel af den dynamiske struktur.

2. Diligence-eksemplet

Et klassisk eksempel i lærebogslitteraturen drejer sog om en forretningsmand, der planlægger en rejse fra sin hjemby A igennem det vilde vesten for at nå frem til sit endelige mål, byen J (hvad han så end skal der). Der er flere alternative ruter fra A til J, og disse ruter involverer en eller flere af byerne B, C, osv. frem til I. Hans problem er, at ikke alle ruter er lige sikre, hvilket i vor model kan omformes til, at omkostninger i form af forsikringer, medbragte gorillaer osv. afhænger af den valgte rute. Lad os antage, at omkostningerne er som vist i tabellen, idet de pladser, hvor der ikke står noget, er ruter, som ikke lader sig gøre eller af anden grund ikke kommer på tale:

	B	C	D	E	F	G	H	I	J
A	22	19	15						
B				16	15				
C				22	16	22			
D					19	22			
E							16	18	
F							18	11	
G							15	13	
H									23
I									29

Lad os se på, hvorledes man kan finde den billigste rute. Vi forkaster på forhånd den primitive metode, der går ud på at checke alle mulige veje fra A til J; selv om det er overkommeligt i dette tilfælde, så vil denne metode være nærmest håbløs, når problemet bliver bare lidt større.

Lidt intuition anvendt på problemstillingen vil antyde en praktisk løsningsmetode. Vi kan forestille os vor rejsende på vej til målet; i hver af de byer, han passerer, står han overfor et beslutningsproblem, nemlig valg af diligencerute videre. Som problemet er skruet sammen, vil der være fire sådanne *trin* af beslutninger undervejs: Først fra A til B, C eller D, så fra en af disse til E, F eller G, og fra disse til H eller I, hvorfra man så når J (det kan man få noget ud af på originalsproget – der er flere "stages" i "stagecoach"-problemet). Den rekursive struktur i beslutningerne kommer frem, når vi noterer os, at omkostningerne ved den optimale politik, når der resterer n trin, og man befinder sig i byen i_n (der er en af dem, som man kan bifinde sig i i $n$te trin), kan skrives som

$$f_n^*(i_n) = \min_{i_{n-1}}\{p(i_n, i_{n-1}) + f_{n-1}^*(i_{n-1})\}, \tag{1}$$

idet $p(j,k)$ er omkostningerne ved turen fra j til k (og for at formalismen kan være komplet sætter vi $f_0^*(J) = 0$).

Det rekursive udtryk i (1) er let at fortolke: Står man ved et vist trin og kender den optimale politik fra næste trin (uanset hvilken by man så er kommet til) og videre frem, så må det bedste man kan gøre nu være at vælge sin tur frem til næste trin så at summen af omkostningerne ved denne tur og de minimale omkostninger videre frem fra den by, man kommer til, er mindst.

Vi kan nu bruge (1) til at løse problemet. I sidste trin før J vi åbenbart

$$f_1^*(H) = 23 + f_0^*(J) = 29 + 0 = 23$$
$$f_1^*(I) = 29 + f_0^*(J) = 29 + 0 = 29.$$

Går vi nu et trin tilbage, kan vi beregne omkostningerne, når der resterer to trin, som

$$\begin{aligned} f_2^*(E) &= \min\{16 + f_1^*(H), 18 + f_1^*(I)\} \\ &= \min\{16 + 23, 18 + 29\} = 39 \end{aligned}$$

og tilsvarende finder vi

$$f_2^*(F) = 40$$
$$f_2^*(G) = 38.$$

Et trin tidligere har vi byerne B, C, D. Vi finder

$$f_3^*(B) = \min\{16 + f_2^*(E), 15 + f_2^*(F)\} = 55,$$

og på samme måde finder vi

$$f_3^*(C) = 56,$$
$$f_3^*(D) = 59.$$

Nu kan vi løse det oprindelige problem, stadig ved hjælp af (1), for i punktet A har vi

$$\begin{aligned} f_4^*(A) &= \min\{22 + f_3^*(B), 19 + f_3^*(C), 15 + f_3^*(D)\} \\ &= \min\{22 + 55, 19 + 56, 15 + 59\} \\ &= 74 \end{aligned}$$

Der altså er laveste mulige omkostninger. Går vi tilbage i beregningerne, ser vi, at i fjerde trin realiseres minimum af byen D; i tredie trin får vi laveste ved at gå til F, og i andet trin går vi fra F til I; disse informationer kunne vi have noteret ned undervejs, så det er let at identificere den optimale beslutning.

Vi har således løst vort komplicerede beslutningsproblem ved at klare en række ret små optimeringsproblemer. Nøglen var udtrykket (1), som er den røde tråd i metoden. Da den også viser sig at være grundideen bag alle andre dynamiske programmeringsproblemer, har man givet den et særligt navn, nemlig *optimalitets-princippet*.. De følgende afsnit viser nogle andre anvendelser af dette princip.

3. Knapsack-problemer

Vi har allerede tidligere set en udgave af et knapsack-problem (det lyder ikke rigtig af noget med "rygsæk"-problem). Her er et eksempel på et sådant:

Antag at vi skal læsse et Hercules-fly med nødforsyninger til et af de efterhånden mange kriseområder. Der ligger fem forskellige sendinger, som vi kan tage med, og for hvert af disse får vi opgivet dels en vægt i tons, dels en værdi målt i kr. (af at få netop denne sending frem). Lad f.eks. disse data være givet som i tabellen:

Sending nr.	1	2	3	4	5
Vægt (w_n)	4	5	7	9	10
Værdi (v_n)	7	12	13	16	19

Vi lader maskinens lasteevne være givet ved T.

Dette er klart nok et (forholdsvis banalt) heltalsprogrammeringsproblem, som vi også kunne have skrevet som

$$\begin{aligned}
&\max 7x_1 + 12x_2 + 13x_3 + 16x_4 + 19x_5 \\
&\text{under bibetingelserne} \\
&4x_1 + 5x_2 + 7x_3 + 9x_4 + 10x_5 \leq T \\
&x_1, x_2, x_3, x_4, x_5 \in \{0, 1\}
\end{aligned}$$

og vi har allerede set på metoder til at løse sådanne problemer. Imidlertid er vi her interesserede i at bruge en fremgangsmåde svarende til forrige afsnits, således at vi deler beslutningsprocessen op i trin og finder optimum ved optimalitetsprincippet.

Trindelingen i denne problemstilling er måske ikke helt oplagt, men lad os bruge den variabel T, som vi allerede har, som udtryk for fri lastekapacitet, og lad os så tage stilling til hver af sendingerne i rækkefølge (først nr.1, så nr.2 osv.). Ved den sidste overvejelse får vi $f_1^*(T)$, værdien af den optimale beslutning ved kapacitet T som enten 0 eller 19: Den er 19 hvis der er plads til den sidste sending, altså hvis $T \geq 10$, og 0 ellers.

Derefter går vi til $f_2^*(T)$, optimal værdi når vi er 2 trin fra slut, og her bruger vi rekursiviteten. For hvert T har vi, at værdien af beslutningen er 19+16 hvis kapaciteten er mindst 10+9, således at der er plads til begge de sidste sendinger, den er 19, hvis der er kapacitet fra 10 op til de 19 (for så kan man have den bedste af de to men ikke begge), den er 16 hvis T er mellem 9 og 10 (så man kan have sending 4 men ikke sending 5), og endelig er den 0 for mindre T, hvor ingen af de to kan komme med. Bemærk, at hver eneste af disse (ret oplagte) sammenhænge er en udgave af vor gamle ven optimalitetsprincippet, nemlig

$$f_2^*(T) = \begin{cases} \max\{f_1^*(T), 16 + f_1^*(T-9)\} & \text{for } T \geq 9 \\ f_1^*(T) & \text{for } T < 9. \end{cases}$$

Går vi et skridt videre, skal vi bruge en tilsvarende relation

$$f_3^*(T) = \begin{cases} \max\{f_2^*(T), 13 + f_2^*(T-7)\} & \text{for } T \geq 7 \\ f_2^*(T) & \text{for } T < 7. \end{cases}$$

og det generelle udtryk ses at have formen

$$f_n^*(T) = \begin{cases} \max\{f_{n-1}^*(T), v_n + f_{n-1}^*(T-w_n)\} & \text{for } T \geq w_n \\ f_{n-1}^*(T) & \text{for } T < w_n. \end{cases}$$

Ved at gå alle skridt tilbage kan man finde den optimale værdi af hele problemet, og hvis man har holdt styr på sine overvejelser undervejs, har man også den optimale politik.

Det ses, at der er en del beregninger knyttet til denne fremgangsmåde (vi ville næppe have valgt metoden hvis det blot var dette problem, det drejede sig om). Fordelen ved dynamisk programmering er, at de enkelte trin meget ofte lader sig simplificere yderligere, således at der bliver tale om en virkelig lettelse af beregningerne.

4. Litteratur

Dynamisk programmering blev til en særskilt disciplin i løbet af 50erne især i kraft af Richard Bellman's bidrag (se f.eks. Bellman og Dreyfus (1962), herunder optimalitetsprincippet. Man kan læse mere herom f.eks. i White (1969).

KAPITEL 8

Grafer

1. Hvad er en graf?

Grafer optræder i en eller anden udgave i de fleste operationsanalytiske problemstillinger. Man støder også på dem i mange andre forbindelser, så

I første omgang vil man intuitivt opfatte en graf som en figur, hvor en række punkter er forbundet med linier. Som sådan har grafen begrænset interesse; det brugbare ved grafer består i, at de er rendyrkede udgaver af problemer, hvor der er abstraheret fra konkrete detaljer.

Grafen er så at sige skelettet i problemet, men som oftest ikke nok til at beskrive problemet fuldt ud. På samme måde giver grafteorien en meget nyttig baggrund til at håndtere de konkrete problemer, men den giver ikke nogen færdig opskrift på problemernes løsning.

Den intuitive diskussion giver os et fingerpeg om, at det der karakteriserer en graf, er dens punkter og forbindelserne imellem dem, men vi mangler endnu en præcis formulering af dette. Det starter vi på her:

En *graf* G består af en mængde V (eller $V(G)$, hvis der er brug for at præcisere, at det er grafen G, der tales om), af punkter, en mængde E (eller $E(G)$) af kanter, og en afbildning fra E til $V \times V$, som til hver kant $e \in E$ angiver to punkter (v_1, v_2) (hvorefter kanten e kaldes en (v_1, v_2)-kant). Disse punkter kaldes for e's endepunkter; e siges at være incident med hvert af punkterne v_1 og v_2 og at forbinde v_1 og v_2.

Vi vil (indtil videre) ikke lægge nogen vægt på rækkefølgen (v_1, v_2), som altså ligeså godt kunne være skrevet (v_2, v_1).

Definitionen giver mulighed for, at der kan være mere end én (v_1, v_2)-kant, for et eller andet par (v_1, v_2) af punkter; i så tilfælde kaldes grafen for en *multipel* graf eller en multigraf. Vi vil dog normalt gå ud fra, at der ikke er sådanne parallele kanter. Der er heller ikke noget i vejen for, at v_1 og v_2 kan være det samme punkt v. En (v, v)-kant kaldes en sløjfe; grafen G kaldes *simpel*, hvis den ikke har sløjfer.

Simple grafer optræder hyppigst i anvendelserne, og det er oplagt, at simple grafer er væsentlig mere overskuelige end generelle grafer. Nedenfor er vist nogle simple grafer med 4 punkter.

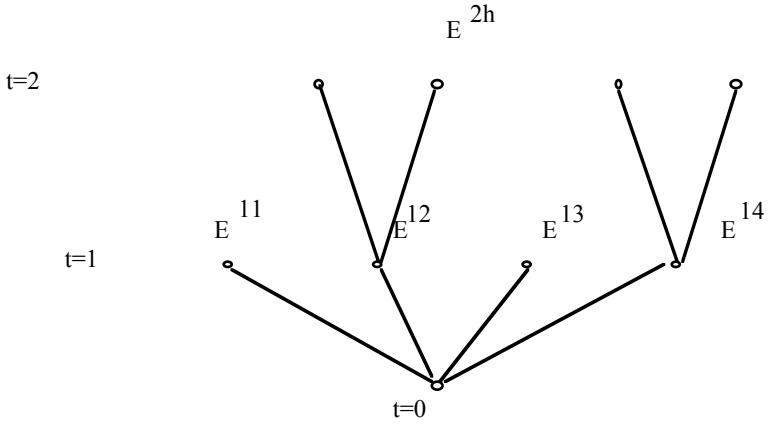
Eksempel 8.1

Lokalisering: En butikskæde overvejer at placere butikker i en landsdel; butikkerne skal placeres i byområder, og det er vigtigt, at kunderne ikke har for lang afstand til en butik.

De givne geografiske omstændigheder (normalt symboliseret ved et kort over området) kan formuleres som en graf, hvor punkterne er byer og linierne er vejforbindelser mellem byerne.

Der er endda lidt mere information: til hver af forbindelseslinierne (kanterne) mellem to punkter er der knyttet et tal, vejlængden mellem de to byer. Vor graf er en såkaldt vægtet graf.

Beslutninger under usikkerhed: En beslutningstager, der skal træffe en beslutning angående fremtidige, usikre forhold, står overfor en situation, der kan illustreres ved grafen nedenfor, et såkaldt træ: Lagene svarer til fremtidige tidspunkter, og forgreningen skabes af de mulige, usikre hændelser, der kan indtræffe ved hvert tidspunkt. Dette beslutningstræ er grundmodellen indenfor teorien om adfærd under usikkerhed.



En variant af ovenstående er spiltræet i et spil på ekstensiv form. Der er også her givet et træ; punkterne i træet fortolkes som "træk" i spillet, og de kanter, der peger opad i figuren, opfattes som valgmulighederne ved det pågældende træk.

Træet hørende til et heltalsproblem: I kapitel 4 så vi, at ved løsningen af et heltalsprogrammeringsproblem ved branch-and-bound metoden var det praktisk at se problemet som et søgeproblem i et træ (figur 2 i kapitel 4).

Vi skal se adskillige andre eksempler på grafer i løbet af dette og de følgende kapitler.

Eksemplerne viser, at der ligger grafer bag mange forskellige slags problemer.

Det bliver hurtigt temmelig uoverskueligt at skrive alle grafer op direkte, selv når det drejer sig om simple grafer. Derfor vil man ofte interessere sig for mindre bestanddele af en given graf; er de små bestanddele først forstået fuldt ud, vil den større sammenhæng ofte være lettere at gennemskue.

En graf $G' = (V(G'), E(G'))$ er en *delgraf* af en anden graf $G = (V, E)$ hvis dens mængder af punkter $V(G')$ og kanter $E(G')$ er delmængder af $V(G)$, respektive $E(G)$. Trivielt er den tomme graf (den, der hverken har punkter eller kanter) og G selv delgrafer af G. En ægte delgraf af G er en delgraf, som ikke er én

af disse to. En delgraf G' af G kaldes *udspændende* hvis $V(G') = V(G)$ (således at G' har samme mængde af punkter som G).

Isomorfi. Den intuitive fremstilling af grafer som punkter med kanter imellem svarer helt naturligt til illustrationerne i forrige afsnit. Det bør dog for ordens skyld bemærkes, at en given graf (givet ved V og E som beskrevet ovenfor) kan afbildes på flere forskellige måder. Der kan også godt i en afbildning af en graf være kanter, som skærer hinanden uden at der hører et punkt til skæringen; de skærer altså ikke hinanden "i virkeligheden", men kun i vor afbildning. Teknisk hænger disse ting sammen med, om vor graf er plan eller ej, noget vi vender tilbage til.

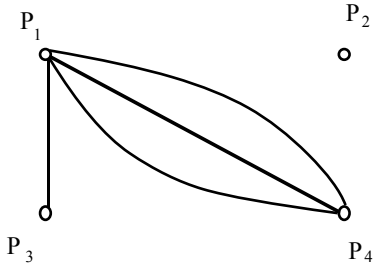
Den præcise udformning af denne observation, at "samme" graf kan afbildes på forskellig måde, får man ved at indføre begrebet *isomorfier* af grafer: To grafer G og G' er isomorfe, hvis der findes bijektive afbildning $\varphi : V(G) \to V(G')$ og $\psi : E(G) \to E(G')$, således at for enhver (v_1, v_2)-kant e er $\psi(e)$ en $(\varphi(v_1), \varphi(v_2))$-kant. I det normale tilfælde (hvor vi ikke har at gøre med multigrafer og hvor grafen er simpel) kan vi nøjes med en lidt simplere version: G og G' er isomorfe, hvis der findes en bijektion $\varphi : V(G) \to V(G')$, således at der for hvert par (v_1, v_2) gælder, at v_1 og v_2 er forbundet med en kant i $E(G)$ netop hvis $\varphi(v_1)$ og $\varphi(v_2)$ er forbundet med en kant i $E(G')$.

Hvis to grafer er isomorfe, vil den ene kunne "omformes" til den anden ved på passende måde at identificere punkter i de to grafer med hinanden. Kanterne mellem punkter, der svarer til hinanden i de to grafer, kan da også sættes i enentyding forbindelse. Grafteorien beskæftiger sig med egenskaber ved grafer, der bibeholdes hos isomorfe grafer.

Blandt de egenskaber ved en given graf (V, E), der ganske åbenlyst bibeholdes under isomorfier, er antallet af punkter i V, antallet af kanter imellem to givne punkter, antallet af kanter ialt, og antallet af kanter, der er incidente med et givet punkt, denne sidste mængde vil vi interessere os lidt mere for i næste afsnit.

Orienterede grafer. En orienteret graf defineres på samme måde som en almindelig graf, nemlig ved en mængde V af punkter, en mængde E af kanter, og en afbildning, der til hver kant e angiver de kanter (v_1, v_2), som kanten forbinder. Forskellen er, at her *har rækkefølgen betydning*. Kanten (v_1, v_2) går fra v_1 til v_2; dette markeres i afbildninger som en pil fra v_1 til v_2. Kanten (v_2, v_1) forbinder de samme punkter, men i modsat retning, og den skal derfor markeres ved en pil fra v_2 til v_1.

Vægtede grafer. Hvis der til en graf (eller en orienteret graf) G er givet yderligere information i form af et tal knyttet til hver kant, har vi en vægtet graf. Tallet kan i anvendelserne være en kapacitet, som angiver den mulige gennemstrømning, en omkostning (ved transport langs kanten), eller en afstand. Vi skal se adskillige eksempler på vægtede grafer.



2. Nogle grafteoretiske begreber

Allerede det forrige afsnit antyder, at grafteori indeholder et væld af definitioner, måske mere end godt er. De bliver selvfølgelig alle brugt i visse sammenhænge, men unægtelig er de fleste af dem ret ligegyldige for vore anvendelser. Vi vil nu alligevel af og til tage et par begreber mere end strengt nødvendigt, blandt andet fordi det sætter tingene ind i en større sammenhæng. De følgende afsnit vil derfor ikke blive brugt rub og stub i fortsættelsen, men der kommer dog en del gods, som der trækkes på senere.

Det er ofte af betydning at se på, hvor mange kanter der går ind i eller ud fra et givet punkt – eller blot hvor mange kanter der berører (er incidente med) punktet. Et udtryk for dette får vi ved at se på et punkts *grad:*

Lad $G = (V, E)$ være en graf og $v \in V$ et punkt i grafen. Ved *graden* (der også optræder under navnet valensen) af v (der skrives som $\deg(v)$ eller $\deg(v, G)$), forstås antallet af kanter i G, som er incident med v, idet dog sløjfer tælles to gange. Hvis der specielt er tale om en orienteret graf, kan vi yderligere definere *indgangsgraden* $\deg_i(v)$ som antallet af kanter med endepunkt i v, og *udgangsgraden* tilsvarende som antallet af kanter med udgangspunkt i v. Vi får dermed, at der må gælde

$$\deg(v) = \deg_i(v) + \deg_o(v)$$

for ethvert punkt v i en orienteret graf.

Graden af et punkt er tydeligt nok et helt tal ≥ 0. Hvis $\deg(v) = 0$, kaldes v for et isoleret punkt. Skrives $\deg(v)$ for punkterne i G op i voksende rækkefølge, fås *gradspektret* for G. I figur 4 er graderne

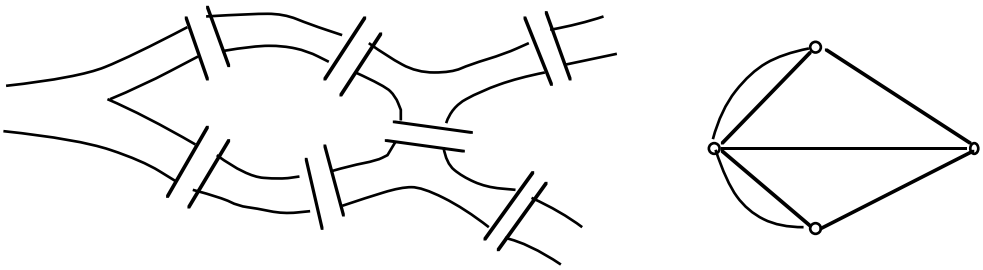
$$\deg(p_1) = 4,\ \deg(p_2) = 1,\ \deg(p_3) = 0,\ \deg(p_4) = 5,$$

så gradspektret er $(0, 1, 4, 5)$.

Såfremt alle punkter i grafen G har samme grad s, kaldes grafen for *regulær* eller, mere præcist, s-regulær. Hvis alle punkter har lige grad, kaldes grafen for en *Euler-graf*.

Eksempel 8.2

Betegnelsen *Euler-graf* skyldes, at begrebet graf første gang er brugt i 1736, hvor det benyttedes af den fra mange andre dele af matematikken kendte *Leonhard Euler* til at løse et bestemt problem:



I byen Königsberg (nu Kaliningrad) var der 7 broer over floden Pregel, og beliggenheden af broerne var som vist på figuren til venstre. Problemet var nu, om det er muligt at gå en tur, således at man passerer hver bro én gang. Dette problem omformulerede Euler ved at se på grafen til højre, hvor der er et punkt for hver bydel og en kant for hver bro; det drejer sig da om, hvorvidt man kan nummerere kanterne sådan, at der for vilkårlige tre successive kanter gælder, at midterste kant har sit ene endepunkt fælles med foregående kant, og sit andet endepunkt fælles med efterfølgende kant. En sådan følge kaldes en *Euler-kredsløb* i grafen, og Euler viste, at der ikke var nogen.

Mere præcist defineres en *tur* i en graf som en endelig følge

$$v_0, e_1, v_1, e_1, v_2, \ldots, v_{k-1}, e_k, v_k$$

af skiftevis punkter og kanter (der ikke nødvendigvis er indbyrdes forskellige), således at e_i er en kant fra v_{i-1} til v_i. Hvis vi har at gøre med en simpel graf, er kanterne helt bestemt af endepunkterne, og vi behøver kun at skrive følgen af punkter. Hvis alle kanter er forskellige, kaldes turen for en *vej;* hvis videre alle punkter (pånær eventuelt v_0 og v_k er forskellige), kaldes vejen for *simpel,* og hvis endelig $v_0 = v_k$, har vi en *simpel lukket vej*. Generelt kalder vi en vej med samme begyndelses- og endepunkt for et *kredsløb*.

Veje i grafer giver anledning til det vigtige begreb *sammenhæng:* En graf $G = (V, E)$ kaldes *sammenhængende* hvis vilkårlige to punkter i grafen er forbundet ved en vej, altså hvis der for alle $v, v' \in V$ findes en vej $(v_0, \ldots, v_m)$ med $v_0, \ldots, v_m \in V$, så at $v_0 = v, v_m = v'$.

Vi kan bruge vort begrebsapparat til at få et generelt resultat om muligheden for Euler-kredsløb. For det første gælder der følgende:

Enhver graf har et lige antal punkter med ulige grad.

Antag nemlig, at vi opregner alle kanter i den givne graf, og for hver af disse kanter opskriver endepunkterne hørende til denne kant. Da vil antallet punkter (ikke nødvendigvis forskellige) opskrevne på denne måde, være to gange antallet af kanter. Graden af et punkt er netop det antal gange, hvor punktet forekommer i

denne opregning af kanternes endepunkter, så vi har sammenhængen

$$\sum_{v \in V(G)} \deg(v) = 2|E(G)|.$$

På højre side står et lige tal, og trækker vi herfra summen over alle punkter med lige grad, har vi stadig et lige tal. Når summen af alle de ulige grader er et lige tal, må der være et lige antal af sådanne punkter. □

Med denne indsigt får vi hurtigt en karakterisering af Euler-grafer:

En graf G uden isolerede punkter (men gerne med multiple kanter) er en Euler-graf hvis og kun hvis den er sammenhængende og alle punkter har lige grad.

Antag først at G er en Euler-graf. Det er oplagt at G må være sammenhængende, for ellers vil der være en kant et eller andet sted, som ikke kan nås i noget kredsløb, endsige i et Euler-kredsløb. Vælges nu et Euler-kredsløb, har vi for hvert punkt på dette kredsløb, at vi kommer ind til det og går ud fra det langs forskellige kanter, så hver gang vi passerer punktet, vokser graden med 2. Da vi passerer alle punkter (et vist antal gange), er graden lige i alle punkter.

For at vise den omvendte sammenhæng, start med et vilkårligt punkt v og lav en vej; vi fortsætter, uden at bruge nogen kant to gange, så længe det kan lade sig gøre. Da alle punkter har lige grad, må vi være kommet tilbage til v, og vi har brugt alle kanter fra v. Hvis der er ubrugte kanter, betragt en delgraf udspændt af disse kanter. Gentag proceduren her. Hvis vi får en lukket vej som har punkter fælles med den første, kan denne udvides til at omfatte det hele. Den længste af alle sådanne lukkede veje bliver et Euler-kredsløb. □

En variation af temaet Euler-grafer er følgende: En graf G siges at være *gennemførlig,* hvis der findes en tur, som gennemløber alle kanter (men som ikke er lukket). Det ses ret let, at en sammenhængende graf er gennemførlig hvis den har netop to punkter med ulige grad.

3. Veje, cykler og træer

Veje er blandt de simplest tænkelige grafer, og interessen for dem hænger sammen med, at mere komplicerede grafer kan have veje af forskellig længde som delgrafer. Vi har allerede set, at veje bruges til at diskutere sammenhæng, og vi skal få brug for dette begreb gentagne gange.

Sammenhængende grafer, hvor hvert punkt har grad 2, kaldes cykler; såvel for veje som for kredsløb betegner *længden* af en cykel antallet af kanter. Bemærk forskellen mellem kredsløb og cykler: I et kredsløb må man gerne besøge et punkt mere end en gang (men dog højest en gang ad hver kant); en cykel besøger kun sine punkter én gang.

Cykler kan også spille en væsentlig rolle som delgrafer af en given graf. Et eksempel på dette er en såkaldt *Hamilton-cykel* i en graf $G = (V, E)$, nemlig en cykel, som indeholder alle punkter i V.

Komplette grafer og turneringer. En *komplet* graf er en simpel graf, hvor hvert par af punkter er forbundet med én kant. Der er klart nok netop én (pånær isomorfi) ikke-orienteret komplet graf med m punkter, og den betegnes med K_m. Hvis en komplet graf er orienteret, kaldes den en *turnering*.

Turneringer optræder i forskellige sammenhænge som udtryk for præferencer, enten individuelle eller kollektive. Hvis punkterne identificeres med alternativer, som er genstand for en vurdering, vil vi forbinde to punkter v_1 og v_2 med en orienteret kant fra v_1 til v_2 netop hvis v_2 anses for bedre end v_1.

Den ordning af alternativerne, der giver anledning til en turnering, må altså være total (alle alternativer kan sammenlignes); den er også asymmetrisk (hvis v_2 er bedre end v_1 kan v_1 ikke samtidig være bedre end v_2). Den er ikke reflexiv (der er ingen sløjfer), og den behøver ikke at være transitiv.

Træer. Et *træ* er en sammenhængende graf, der er *acyklisk* i den forstand, at den ikke indeholder cykler. Træer er nok den type af grafer, der har fundet de fleste anvendelser, ikke blot i økonomi og beslutningsteori, men også indenfor f.eks. datalogi og fysik.

Der er flere forskellige måder at karakterisere et træ på:

Lad $G = (V, E)$ være en graf uden sløjfer, og antag at der ialt er m punkter og n kanter i G. Da er følgende udsagn ækvivalente:

(i) G er et træ,

(ii) For et vilkårligt par (v, v') af punkter i G er der netop én vej, som forbinder v og v',

(iii) G er sammenhængende, og $m = n + 1$,

(iv) G er acyklisk, og $m = n + 1$.

Beviset er ikke så indviklet, men det er ret langt, da sætningen selv indeholder ganske mange udsagn. For at hjælpe på overblikket er der givet en populær forklaring af, hvad der sker, i firkantede parenteser.

(i) $\Rightarrow$ (ii). [Hvis der var to veje mellem to forskellige punkter, så kunne man gå frem ad den ene vej, hjem igen ad den anden, og dermed havde man en cykel, en modstrid] Antag at G er et træ og altså sammenhængende, således at vilkårlige to punkter er forbundet ved en vej, men at der er to punkter v og v', der kan forbindes med forskellige veje

$$(v_0^1, \ldots, v_{m_1}^1) \text{ og } (v_0^2, \ldots, v_{m_2}^2),$$

hvor

$$v_0^1 = v_0^2 = v,\ v_{m_1}^1 = v_{m_2}^2 = v'.$$

De to veje starter altså i samme punkt; lad i^0 være det største index punkt med den egenskab, $v_i^1 = v_i^2$ for $i \leq i^0$, og lad i^1 være det mindste index $> i^0$ så at $v_{i^1}^1$ også er et punkt i vejen $(v_0^2, \ldots, v_{m_2}^2)$, dvs.

$$v_{i^1}^1 = v_{i^2}^2$$

for et vist index i^2. De to veje $(v_{i^0}^1, \ldots, v_{i^1}^1)$ og $(v_{i^0}^2, \ldots, v_{i^2}^2)$ udgør da en cykel. Men dette strider mod at G er et træ, så der kan kun være én vej mellem vilkårlige to punkter.

(ii) $\Rightarrow$ (iii). [Byg grafen op fra bunden ved at føje punkter til og tegne kanterne efterhånden som det kan gøres. Da må man nødvendigvis hele tiden være en kant i underskud] Da vilkårlige to punkter er forbundet med en vej, er G sammenhængende. Vi mangler altså bare at vise, at $m = n + 1$, og hertil bruger vi induktion efter m, antallet af punkter. Hvis $m = 1$ kan der ikke være nogen kanter, for en sådan ville være en sløjfe; altså er $n = 0$ og resultatet holder.

Antag nu, at resultatet holder for alle grafer med færre end m punkter, som opfylder (ii), og lad G være en graf med m kanter, der opfylder (ii). Vælg en vilkårlig kant, f.eks. (v_1, v_2)-kanten e. Da er e den eneste vej mellem v_1 og v_2. Vi danner nu grafen $G - e$ (som har samme mængde af punkter som G og samme kanter pånær e); hvis vi ser på de to delgrafer af $G - e$, som udspændes henholdsvis af alle de punkter, der kan forbindes ved en vej med v_1, og af alle de punkter, der kan forbindes med en vej med v_2, ses disse to delgrafer at indeholde alle G's punkter, og at opfylde betingelsen i (ii). Den samlede mængde af kanter i disse to delgrafer må ifølge induktionsantagelsen være $m - 2$. Lægges hertil kanten e fås, at antallet af kanter i G er $m - 1$.

(iii) $\Rightarrow$ (iv). [Antag at der var en cykel. Den har lige så mange punkter som kanter. Altså er der punkter som ikke hører til den. Men sådanne punkter må være forbundet med cyklen, og igen får vi lige så mange punkter som kanter, en modstrid] Lad G være en graf med m punkter, som indeholder en cykel. Lad $C = \{v_1, \ldots, v_k\}$ være punkterne i cyklen.

Vi kan antage, at C er valgt maximal i den forstand, at der ikke er nogen cykel i G, som har C som ægte delmængde af sin punktmængde. Det betyder, at for ethvert $v \notin C$ er der højest én kant, som forbinder v med et punkt i C. Lad C_1 være mængden af punkter, der er forbundet med C ved en (entydig) kant.

Fortsættes på denne måde, fås i h'te trin en mængde C_h af punkter, der kan forbindes ved en kant med punkter i C_{h-1}; dette giver en entydig tilordning af kanter til punkter (ellers ville C_0 ikke være maximal). I endelig mange skridt fås, at alle punkter i V kan tilordnes en kant. Da denne tilordning er entydig, må der være lige mange punkter og kanter i G, en modstrid. Vi konkluderer, at G er acyklisk.

(iv) $\Rightarrow$ (i) Antag, at G ikke er sammenhængende; da må G bestå af et vist antal sammenhængende delgrafer (for hvert punkt v har vi en sammenhængende delgraf bestående af alle de punkter $v' \in V$, for hvilke der findes en vej i G forbindende v

og v'). Hver af disse delgrafer er acyklisk (da G er det), dvs. et træ. Vi benytter nu (iii) på delgraferne og får, at antallet af kanter i hver af dem er lig antallet af punkter minus 1. Men så kan der ikke være mere end én delgraf, dvs. G må være sammenhængende. □

I mange situationer vil der i et træ være et bestemt punkt udpeget til at spille en særlig rolle i fortolkningerne. Et sådant punkt kaldes for *roden* i træet. Hvis grafen ikke er orienteret, kan roden vælges vilkårligt. I en orienteret graf, som også er et træ, er roden entydigt bestemt ved at indgangsgraden er 0.

4. Afstand i grafer; farvning

Lad G være en sammenhængende graf. Hvis v_1 og v_2 er punkter i G, findes der altså (mindst) en vej, som forbinder v_1 og v_2. Vi kan da definere *afstanden* mellem v_1 og v_2 som længden af den korteste vej mellem v_1 og v_2. Vi betegner denne afstand med $d(v_1, v_2)$; det er let at se, at $d(\cdot, \cdot)$ opfylder

$$d(v_1, v_2) = d(v_2, v_1),$$
$$d(v_1, v_3) \leq d(v_1, v_2) + d(v_2, v_3),$$
$$d(v_1, v_2) = 0 \Leftrightarrow v_1 = v_2.$$

Vælges et punkt v_0 i en sammenhængende graf $G = (V, E)$, kan man definere *afstandsklasser* i V ved

$$\mathcal{D}_s = \{v \in V | d(v_s, v_0) = s\},\ s = 0, 1, \ldots.$$

Mængderne $\mathcal{D}_0, \mathcal{D}_1, \ldots$ er indbyrdes disjunkte, og ethvert punkt i V ligger i én af disse mængder (de udgør med andre ord en klassedeling af V). Videre må alle punkter i $\mathcal{D}_s$ være forbundet med mindst ét punkt i $\mathcal{D}_{s-1}$ (længden af den korteste vej fra et sådant punkt til v_0 er s, så fjernes den første kant, må der være $s - 1$ tilbage). På den anden side kan der ikke være kanter, der forbinder $\mathcal{D}_s$ med $\mathcal{D}_t$ hvis $|t - s| > 1$.

Når man har et afstandsmål i en graf $G = (V, E)$, kan man også finde den største afstand mellem to punkter i grafen, nemlig grafens *diameter*

$$\delta(G) = \max_{v, v' \in V} d(v, v').$$

Afstandsmålet virker umiddelbart fjernt fra, hvad man intuitivt ville forstå ved "afstand", men det er et nyttigt teknisk hjælpemiddel. Vi har faktisk allerede brugt det implicit i beviset for sætningen ovenfor. Her er et andet resultat, der kan vises ved at bruge afstandsklasser:

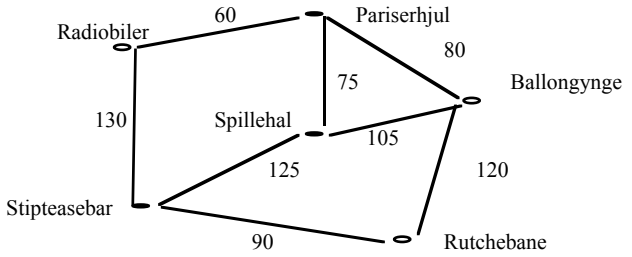
Enhver sammenhængende graf har et udspændende træ.

Eksempel 8.3

Minimalt udspændende træ. I en forlystelsespark skal der lægges elforsyning mellem de enkelte attraktioner, som er vist i figuren nedenfor sammen med deres indbyrdes afstande.

Tilslutningen udefra kan ske hvorsomhelst, og installationen ønskes lavet på en sådan måde, at man bruger mindst mulig kabel. Hvorledes finder man frem til den billigste løsning?

Vi vil antage, at installationen skal følge de ruter, som er vist på figuren (man kan altså ikke lægge kabel tværs over plænerne eller bygge en fordelerstation et vilkårligt sted). Dermed reducerer problemet til at finde en sammenhængende graf, som har de samme punkter som den oprindelige (som er udspændende iflg. vor terminologi fra afsnit 3). Da den skal være minimal i den forstand, at der ikke er flere kanter med end højst nødvendigt, får vi, at løsningen bliver et træ. Men hvilket?



Det findes der en nem algoritme til at beregne: Man går frem som følger:

Trin 1. Vælg et vilkårligt punkt; i eksemplet vælger vi ballongyngen.

Trin 2. Vælg det endnu ikke forbundne punkt, som er nærmest ved et allerede forbundet punkt, og forbind med dette. I vort eksempel vælger vi pariserhjulet.

Trin 3. Hvis der stadig er ikke-forbundne punkter tilbage, fortsættes med trin 2; i modsat fald har vi en optimal løsning.

I eksemplet fortsætter vi med radiobilerne, derefter spillehallen, og til sidst rutchebanen og stripteasebaren.

Det er ikke af betydning, hvor der startes; man vil under alle omstændigheder få den samme løsning (eksemplet skyldes S.S.Cohen, 1985).

Vælg et punkt v_0 og definer afstandsklasserne $\mathcal{D}_0, \mathcal{D}_1, \ldots$ som ovenfor. Vi vil lave en ny graf G' ved at fjerne kanter fra G indtil vi ender med et træ; det sker på følgende måde:

Fjern først alle kanter mellem punkter i samme afstandsklasse. For $s \geq 1$ og for ethvert $v \in \mathcal{D}_s$ fjernes alle kanter mellem v og et punkt i $\mathcal{D}_{s-1}$. Ifølge vore bemærkninger ovenfor er ethvert punkt i $\mathcal{D}_s$ forbundet ved en kant med et punkt i $\mathcal{D}_{s-1}$, så ethvert v må være forbundet ved en vej med v_0. Altså er vor G' sammenhængende; det er klart, at G' udspænder G.

Vi mangler nu blot at vise, at G' er acyklisk. Antag, at der er en cykel i G‘, og lad t være det største tal så at cyklen har et punkt $\tilde{v}$ i $\mathcal{D}_t$ (den kan ikke have mere end ét punkt, for der er ingen kanter mellem punkter i samme afstandsklasse). Der må da være to forskellige kanter, der forbinder $\tilde{v}$ med punkter i $\mathcal{D}_{t-1}$ i strid med konstruktionen af G'. Vi konkluderer, at G' er acyklisk og altså et træ. □

En graf $G = (V, E)$ kaldes *todelt* hvis V kan opdeles i to disjunkte mængder V_1 og V_2 således at enhver kant i G forbinder et punkt i V_1 og et punkt i V_2. En todelt graf er komplet hvis ethvert par (v_1, v_2) med $v_1 \in V_1, v_2 \in V_2$ er forbundet med en kant. Den komplette todelte graf med m_1 punkter i den ene klasse og m_2 i den anden betegnes K_{m_1,m_2}. Grafen $K_{m,1}$ kaldes en *stjerne* med $m + 1$ punkter.

Der gælder følgende:

En sammenhængende graf $G = (V, E)$ er todelt hvis og kun hvis enhver cykel i G har lige længde.

"Kun hvis". Skrives cyklens punktmængde som $(v_0, \ldots, v_m)$ med $v_m = v_0$, må hver anden (svarende til index $0, 2, 4, \ldots$) være fra V_1 og hver anden $(1, 3, \ldots)$ fra V_2. Det er klart, at m er lige.

"Hvis". Vælg et punkt v_0 og lad $\mathcal{D}_0, \mathcal{D}_1, \ldots$ være de tilhørende afstandsklasser. Lad

$$V_1 = \cup_{s=0,2,4,\ldots}\mathcal{D}_s,\ V_2 = \cup_{s=1,3,5,\ldots}\mathcal{D}_s.$$

Hvis to punkter v, v' i enten V_1 eller V_2 er forbundet ved en kant, da må de tilhøre samme afstandsklasse. For hver af de to punkter vælges en korteste vej til v_0. Hvis $\tilde{v}$ er det fjerneste (fra v_0) punkt som er på begge veje, da har vi en cykel i G. Dens længde er sammensat dels af vejene fra v og fra v' til $\tilde{v}$, som er lige lange, dels af kanten mellem v og v'. Cyklen har altså ulige længde, en modstrid. Vi konkluderer, at grafen er todelt. □

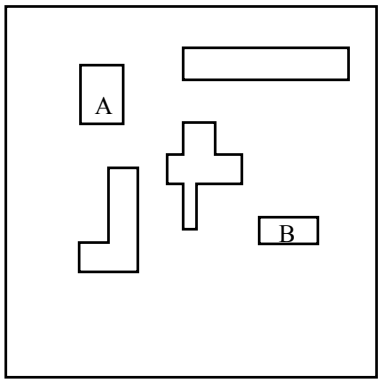
Todelte grafer er et specielt tilfælde af noget mere generelt, nemlig den såkaldte farvning af grafer. En *k-farvning* af en graf $G = (V, E)$ er en klassedeling af V i k delmængder (farveklasser) $V_1, \ldots, V_k$, således at der for et vilkårligt par (v_1, v_2) af punkter gælder, at hvis der er en kant mellem v_1 og v_2, da ligger v_1 og v_2 i forskellige farveklasser. Det mindste tal k for hvilket der findes en k farvning af G kaldes for G's *kromatiske tal* og betegnes $\chi(G)$. Et berømt problem i matematikkens historie er det såkaldte 4-farve problem, der har været diskuteret siden midten af forrige århundrede. Kort fortalt er problemet, om det er nok at bruge 4 farver til at farve et landkort på en sådan måde, at lande, der grænser op til hinanden, har forskellig farve.

Omformuleret til vores terminologi har vi at gøre med *plane* grafer, dvs. grafer, der kan tegnes i planet således at ingen kanter skærer hinanden. Her svarer punkterne til lande, og der er en kant mellem to punkter netop hvis landene har fælles grænse. Problemet er dermed: Har alle plane grafer kromatisk tal 4?

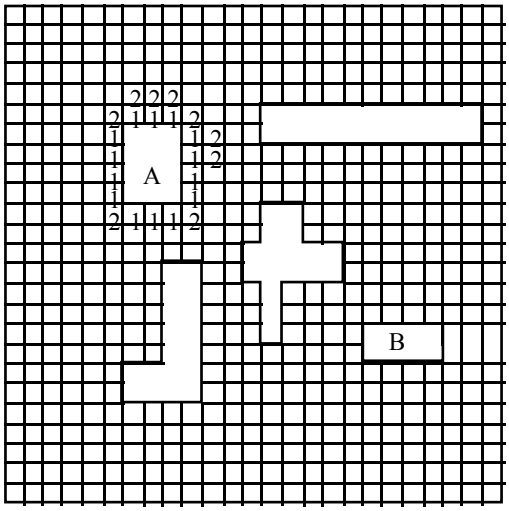
Trods den tilsyneladende "anvendelsesorienterede" historie bag dette problem har det næppe stor praktisk betydning, men der er en omfattende litteratur om det. En løsning på problemet blev givet i 1977 af Appel og Haken. Beviset omfatter

Eksempel 8.4

Et praktisk optimeringsproblem, der benytter afstandsklasser i en graf, er følgende: En elektronikvirksomhed skal fremstille en printplade, og i denne forbindelse er det nødvendigt at føre en ledning fra A til B på pladen nedenfor. De forskellige figurer indtegnet svarer til en række faste komponenter på pladen, som ledningen skal føres udenom. Det er praksis at føre ledninger i rette linier parallelt med printpladens sider. Hvordan skal ledningen føres, når der ønskes så kort en ledning som muligt?



En brugbar fremgangsmåde er at starte med at inddele pladen i et tilstrækkelig fintmasket net. Vi identificerer kvadraterne i nettet med punkter i en graf, og der er kanter imellem dem, hvis kvadraterne støder op mod hinanden. Det er nu helt ligetil at indlægge afstande fra A som vist på figuren, og når man når B, har man samtidig løst problemet, idet man blot skal vælge en vej, så at man hele tiden går til højere afstandsklasse.



Eksemplet kan måske umiddelbart se trivielt ud, men det var jo også lavet som et overskueligt eksempel. Hvad det gerne skulle antyde – og hvad der faktisk passer – er at grafteori har fået et meget stort anvendelsesområde i forbindelse med algoritmer til løsning af praktiske problemer, som måske ikke i sig selv ser indviklede ud, men som sandelig bliver det, når størrelsen af problemet vokser tilstrækkelig meget.

imidlertid flere hundrede sider tekst og kræver timevis af kørsel på maskine, så det er ikke helt let at gå til - og checke.

5. Grader af sammenhæng i grafer

Det centrale i formuleringen af et eller andet problem i grafteoretiske termer er som regel studiet af, hvorledes punkterne i grafen "hænger sammen", løst sagt hvor mange forskellige måder man kan komme fra det ene til det andet punkt i grafen på.

Vi har allerede set en del til sammenhængende grafer. Vi har også noteret os, at en graf, der ikke er sammenhængende, kan omfattes som sammensat af sammenhængende delgrafer. Vi kalder disse for grafens *sammenhængskomponenter.* Vi kan alternativt beskrive grafens grad af sammenhæng ved at se på, hvor meget der skal fjernes før vi får en sammenhængende graf.

For at diskutere de to betragtningsmåder lidt mere præcist vil vi indføre henholdsvis punkt- og kantsnit: Lad $G = (V, E)$ være en graf; en delmængde W af V kaldes et *punktsnit* i G hvis der er mindst én sammenhængskomponent $G' = (V', E')$ i G således at grafen $G' - (W \cap V')$ ikke er sammenhængende. Tilsvarende kaldes en mængde $F \subset \mathcal{E}$ af kanter i G er et *kantsnit* hvis der er mindst én sammenhængskomponent $G' = (V', E')$ i G så at $G' - (F \cap \mathcal{E}')$ er ikke-sammenhængdende.

Vi indfører nu tallet $\kappa(G)$ som det mindste tal t for hvilket G har et punktsnit bestående af t punkter. Dette er vort første mål for graden af sammenhæng, idet det netop angiver, hvor mange punkter der må fjernes for at ændre på den sammenhængsstruktur (givet ved sammenhængskomponenterne), som grafen var i besiddelse af.

Den alternative betragtningsmåde fås ved at se på antallet af forskellige veje mellem to punkter v_1 og v_2. Her vil vi for at opfatte to veje som forskellige kræve, at de ikke har noget punkt tilfælles bortset fra endepunkterne. To sådanne veje kaldes disjunkte, og vi kan nu definere $\sigma(G)$ som det største tal s således at der for ethvert par (v_1, v_2) af punkter i V findes s parvis disjunkte veje mellem v_1 og v_2.

Hvis W er et punktsnit i G, og punkterne v_1 og v_2 ligger i forskellige sammenhængskomponenter i $G - W$, da må hver af de $\sigma(G)$ parvis disjunkte veje mellem v_1 og v_2 (i G) indeholde et punkt fra W, hvilket igen betyder, at W har mindst $\sigma(G)$ elementer. Vi har dermed vist, at

$$\sigma(G) \leq \kappa(G);$$

Der gælder imidlertid et stærkere resultat:

Menger's sætning: For enhver graf $G = (V, E)$ gælder, at

$$\sigma(G) = \kappa(G).$$

Beviset for Menger's sætning er ikke helt ligetil, men det kan fås på en simpel måde ved at bruge den såkaldte max-flow-min-cut sætning, som vi viser i næste

kapitel. Når de to mål giver samme resultat, kan vi nøjes med et af dem; man plejer at bruge $\kappa(G)$, som kaldes *konnektiviteten* af G, og man siger, at G er t-sammenhængende for ethvert $t \leq \kappa(G)$.

Vi har i det foregående arbejdet med punktsnit; vi kunne også alternativt have interesseret os for kantsnit: Vi skulle da definere tal $\lambda(G)$ som det mindste t så at G har et kantsnit bestående af t kanter, og tallet $\mu(G)$ som det største tal s for hvilket to punkter i G er forbundet med s parvis kantdisjunkte veje (hvor kantdisjunkt defineres på den oplagte måde). Menger's sætning i kant-udgave siger da, at $\lambda(G) = \mu(G)$; den fælles værdi, der betegnes $\lambda(G)$, kaldes *kantkonnektiviteten* af G, og G siges at være t-sammenhængende for alle $t \leq \lambda(G)$.

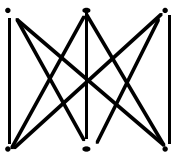
6. Plane grafer

Vi har tidligere nævnt, at en graf siges at være plan, hvis den kan tegnes i planet således at intet par af kanter skærer hinanden udenfor punkterne. Plane grafer har visse direkte anvendelser – og giver start til overvejelser, som fører meget længere.

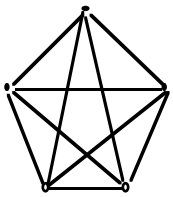
Eksempel 8.5

Tre huse skal have lagt ledninger til henholdsvis gas, vand og el. Det antages, at ledningerne ikke må krydse hinanden (og vi ser bort fra muligheden for at grave dem ned i forskellig dybde).

Den graf, der opstår, når vi tegner alle forbindelser mellem de tre huse og de tre værker, kan hurtigt identificeres som $K_{3,3}$, den perfekte todelte graf med to gange tre punkter. Denne graf kan f.eks. illustreres som vist i figur 12:



$K_{3,3}$



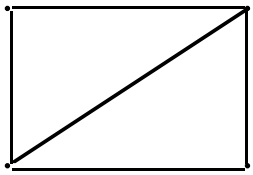
K_5

Den illustrerede graf overtræder ganske vist kravet om, at kanter ikke må skære hinanden, men det kan jo tænkes, at der er en anden måde at lægge ledninger på, så at det lykkes.

Et par hurtige forsøg på at trække ledninger vil imidlertid overbevise én om, at det kniber. Hvordan man end starter, ender man med mindst en ledning, der ikke kan komme igennem til den ønskede modtager uden at skære en anden. Vi har nu $K_{3,3}$ stærkt mistænkt for ikke at være plan.

En lidt mere systematisk behandling af problemet om, hvornår en graf er plan, går den anden vej og starter med en plan graf. En *region* i en plan graf er et sammenhængende område i planen, som bliver tilbage når grafens kanter og punkter er fjernet. De punkter og kanter, der støder op til en region, siges at være incidente

med regionen og kaldes dens rand. Grafen i figuren nedenfor har 3 regioner, nemlig de to trekanter inde i grafen og den ydre, omgivende region.



Vi skal først bruge et par hjælperesultater:

(1) Lad G være en sammenhængende plan graf med p punkter, q kanter, og r regioner. Da gælder der

$$p - q + r = 2.$$

Vi bruger induktion efter q. Hvis $q = 0$ består grafen af et eneste punkt, og der er kun en region, nemlig den ydre, så sætningen passer.

Antag nu, at resultatet holder, når $q \leq k - 1$, og lad G have k kanter. Hvis G er et træ, ved vi allerede fra sætning ?, at $p = k + 1$, og det er oplagt, at $r = 1$, så $p - k + r = (k + 1) - k + 1 = 2$. Hvis ikke G er et træ, har G en cykel; vælg en kant e på denne, og betragt grafen $G - e$. Denne graf er også sammenhængende, netop fordi vi fjernede en kant fra en cykel i G.

Der er to regioner i G, der er incidente med e (de to sider af e), og disse to regioner bliver til én i $G - e$. Derfor har $G - e$ $k - 1$ kanter og $r - 1$ regioner, men samme antal punkter som før. Vi bruger nu induktionsantagelsen på $G - e$ og får

$$2 = p - (k - 1) + (r - 1) = p - k + 1,$$

hvilket var, hvad vi skulle vise. □

Resultatet er en klassiker i grafteorien, men har langt ældre rødder (Euler) og rækker ind i en masse anden matematik. Vi kan bruge den til et andet resultat:

Lad G være en plan graf med p punkter og q kanter, hvor $p \geq 3$. Da gælder

$$q \leq 3p - 6.$$

Resultatet holder for $p = 3$, for en graf med 3 punkter kan ikke have mere end 3 kanter. Vi kan derfor antage, at $p \geq 4$. Betragt nu en afbildning i planen, og betegn antallet af regioner med r. For hver region R bestemmer vi antallet af kanter i R's rand, hvorefter vi summer over regioner. Lad N være summen; da hver region må have mindst 3 kanter på sin rand, har vi $N \geq 3r$. På den anden side vil vi

aldrig kunne tælle en kant med mere end højst to gange, for der er kun to regioner, for hvilke en given kant optræder på randen; vi har derfor $N \leq 2q$. Ialt får vi

$$3r \leq N \leq 2q$$

eller $-r \geq -2q/3$. Nu bruger vi sætning 7, som siger os, at

$$p = q - r + 2 \geq q - \frac{2q}{3} + 2 = \frac{q}{3} + 2,$$

hvoraf vi får det ønskede resultat. □

Nu har vi et værktøj, der kan afsløre det for os, hvis en graf ikke er plan. Vi skal simpelthen tælle punkter og kanter og checke, om sammenhængen ovenfor er opfyldt. Er den ikke det, kan vi godt opgive at afbilde den i planen.

Et klassisk eksempel på en graf, der ikke er plan, er K_5, den perfekte graf med 5 punkter. Den har 5 punkter og ligeså mange kanter, som der er par af forskellige punkter, dvs. $\binom{5}{2} = 10$ kanter. Imidlertid er $3p - 6 = 15 - 6 = 9 < 10$.

Vi kan også, omend knap så umiddelbart, få en "løsning" på problemet om de tre huse og tre værker: Grafen $K_{3,3}$ er ikke plan. Antag nemlig, at den var, og tegn en plan udgave af den. Vi lader igen N være summen over alle r regioner af antallet af regionens kanter. Da $K_{3,3}$ er todelt, har grafen ingen trekanter (overvej!), så $N \geq 4r$. På den anden side har vi at N tæller hver kant højst to gange, så $N \leq 2q$, hvor q er antallet af kanter i $K_{3,3}$, som er lig med 9 (tag den ene af de to mængder i todelingen; fra hvert af dens tre punkter udgår tre kanter, og dette er netop samtlige kanter i grafen). Ialt har vi $4r \leq 18$ eller $r \leq 9/2$. Bruger vi nu sætning ? får vi $6 - 9 + r = 2$ eller $r = 5$, og det er en modstrid.

Vi har nu to eksempler på grafer, nemlig K_5 og $K_{3,3}$, som ikke er plane. Den kan udbygges, idet der selvfølgelig må gælde, at en graf, der har enten K_5 eller $K_{3,3}$ som delgraf, ikke kan være plan. Men vi mangler stadig en generel regel. Den viser sig dog at hænge ret nøje sammen med, hvad vi allerede har. Vi skal blot bruge endnu et begreb (alene til dette formål): En *underdeling* af en graf G er en graf opnået fra G ved at indsætte punkter (af grad 2) på kanter i G. En graf anses for en (triviel) underdeling af sig selv.

Det kan vises (men det vil vi ikke gøre), at der gælder følgende (Kuratowski's sætning):

Kuratowski's sætning: En graf G er plan hvis og kun hvis G ikke har nogen delgraf, som er isomorf med en underdeling af K_5 eller af $K_{3,3}$.

7. Litteratur

Grafteori er både ny og gammel, for de problemer, der behandles, har man diskuteret i århundreder, mens der på den anden side først opstod en samlet, selvstændig

disciplin omkring århundredeskiftet. Blandt de internationalt kendte grafteoretikere fra denne begyndelse er iøvrigt den danske matematiker Julius Petersen. En klassiske lærebog i matematisk grafteori er Harary (1969), og en simpel introduktion er Chartrand (1985).

Grafteorien har fået en vældig opblomstring i den seneste tid, fordi den indgår som et uundværligt element i konstruktionen af computer-algoritmer og analysen af alternative algoritmers fortrin og ulemper. En fremstilling af grafteorien udfra dette grundsynspunkt kan findes i McHugh (1990).

KAPITEL 9

Netværk

1. Indledning

I en lang række anvendelser optræder grafer som den underliggende struktur i problemstillingen, men der er foruden grafen også andre data af betydning. Da grafen ofte symboliserer de mulige måder at flytte sig fra en lokalitet eller tilstand til en anden, vil den oplagte ekstra information være data om, *hvor meget* der kan flyttes ad hver mulig vej.

Når information af denne type føjes til en graf, foreligger der et *netværk*. Der er grund til at kigge lidt nærmere på netværk, for der er temmelig mange økonomiske problemstillinger, som med fordel lader sig formulere som sådan; ofte er det endda helt naturligt, at der foreligger netværksproblemer. Hvis det drejer sig om at forsyne kunder fordelt geografisk over et større område med varer ad visse, givne ruter på mest hensigtsmæssige måde, så er sagen allerede serveret som et netværksproblem. Dertil kommer – hvad der nok er det afgørende – at der findes særlige metoder skræddersyet til netværksproblemer. Egentlig åbner disse metoder ikke fundamentalt nye muligheder, for som regel kunne de problemer, vi behandler, alternativt være formuleret som lineær programmering, men de giver en genvej, som er nyttig, især når problemet får en passende størrelse (det sidste kan vi ikke rigtig nyde, givet at vi under alle omstændigheder nøjes med ret små problemer, hvor vor metode ikke virker påfaldende hurtig; men i den større sammenhæng er det en forbedring). Endelig kommer der også nogle biprodukter ud, der i højere grad har teoretisk interesse.

2. Definition af netværk

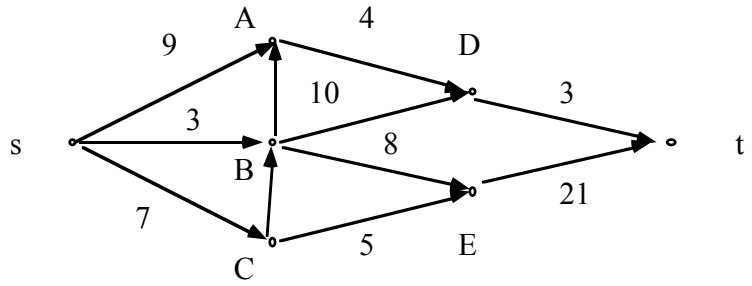
Ved et (orienteret) netværk $\mathcal{N}$ forstås en orienteret graf

$$\Gamma = (V, E)$$

med to særlige punkter $s \in V$ og $t \in V$, kaldt henholdsvis netværkets *kilde* og *terminal,* hvor der til hver kant e er givet et tal $c_e \in \mathbf{R}_+$, *kapaciteten* af e.

Et eksempel på et netværk er vist i figur 1. Hver kant i den orienterede graf har fået et tal knyttet til sig. Hvis grafen symboliserer mulige veje mellem

punkterne (der f.eks. kan være byer), da er kapaciteten et udtryk for den maximale gennemstrømning i kanten indenfor et givet tidsrum. Den simpleste fortolkning af netværket er den, der allerede er antydet i betegnelserne. Vi forestiller os, at der er ophobet en større godsmængde i kilden, og at vi gerne vil have overført så meget som muligt til terminalen; kapaciteterne af de enkelte kanter angiver, hvor meget der kan presses dem igennem pr. tidsenhed. Problemet er da at finde en samlet transportplan, der udnytter alle de kanter, der kan bruges, for at få flyttet så meget som muligt. Hvor meget er det, og hvordan vil transporten faktisk foregå?



Figur 1

Dette skal præciseres en smule. Ved en *strøm* (eng.: flow) i netværket $\mathcal{N}$ forstå en specifikation af, hvad der bevæger sig ad hver eneste rute, dvs. langs hver kant, i netværket. Skrevet op bliver dette til en familie $f = (f_e)_{e \in E}$ af tal, hvor indekset e løber over alle kanterne i netværket, og som opfylder følgende to betingelser (der er ganske naturlige, når først man indser, hvad de udtrykker):

(1) For hvert punkt v i $\mathcal{N}$ forskelligt fra s og t, gælder der

$$\sum_{e=(v,v')} f_e = \sum_{e=(v'',v)} f_e,$$

(dvs. strømmen ind i punktet v svarer til strømmen ud af v, der bliver ikke noget liggende i v);

(2) For hver kant e gælder der, at

$$0 \leq f_e \leq c_e$$

(strømmen i en given kant er ikke-negativ og kan ikke overskride kapaciteten).

Den første af betingelserne går normalt under betegnelsen "flow conservation" og sikrer, at der ikke bliver noget væk undervejs: Alt hvad der går ind i et eller andet punkt (bortset naturligvis fra terminalen) går også ud igen. Naturligvis er det ikke meningen med denne betingelse at postulere, at sådan er alle transportsystemer. Det kan da udmærket tænkes, at der i praktiske anvendelse sløses noget væk undervejs, men så er den del af sagen ikke et netværksproblem. Her kan det betale sig at rendyrke det essentielle, nemlig at finde den størst mulige strøm gennem netværket.

Den anden betingelse er ganske banal. Vi skal sidst i kapitlet fundere lidt på, hvad vi kan stille op hvis der er flere restriktioner end disse på, hvad der kan gå igennem de enkelte kanter.

For en given strøm, der som nævnt kan skrives som $f = (f_e)_{e \in E}$, kan vi angive den samlede mængde, som er flyttet fra s til t: Da der ikke bliver noget liggende undervejs, må det svare til, hvad der er flyttet ud af s, dvs.

$$V(f) = \sum_{e=(s,v')} f_e - \sum_{e=(v'',s)} f_e.$$

Vi kunne også godt have opgjort $V(f)$ som alt det, der sendes ind i t, for det bliver jo det samme på grund af flow conservation. Der ville endda være flere andre måder, som vi skal se senere.

En strøm er *maximal* hvis dens værdi er størst blandt alle mulige strømme i det givne netværk. Som det kunne forventes, er det centrale problem i dette kapitel er at finde en maximal strøm i et netværk. Det vil vi gå i kast med nu, idet vi starter med nogle mere generelle overvejelser. De ser ikke ud af så meget, men de er til hjælp når vi opstiller den generelle algoritme.

3. Maximale strømme

Da man jo ikke ved simpel inspektion af et netværk kan afgøre, om en given strøm er maximal eller ej, må der bruges en systematisk metode, en algoritme, og vi skal i det følgende beskæftige os sådanne algoritmer. Der findes flere forskellige, men ihvertfald de mest brugte er skruet sammen på et fælles grundlag, og det starter vi med. Vort første skridt vil være at se på, hvornår vi kan finde strømme med større værdi end en given strøm.

Lad $\mathcal{N}$ være et givet netværk, og lad f være en strøm i $\mathcal{N}$. Det er oplagt, at hvis der findes en orienteret vej fra s til t – som vi kunne kalde $\mathcal{P}_1$; det er blot en række kanter lagt i fortsættelse af hinanden, således at orienteringerne passer, hver kant peger på starten af den efterfølgende – således at strømmen f_e i hver kant e på vejen er mindre end kantens kapacitet c_e, da kan vi finde en ny strøm f' med større værdi; vi kan nemlig lægge

$$\triangle f = \min\{c_e - f_e | e \in \mathcal{P}_1\}$$

til hvert f_e for e en kant på vejen $\mathcal{P}_1$, og vi får da

$$V(f') = V(f) + \triangle f.$$

I denne situation er det altså temmelig ligetil at forbedre resultatet.

Et tilsvarende tilfælde af forbedring er det, hvor der findes en orienteret vej $\mathcal{P}_2$ fra t til s (altså en "forkert" rettet vej), hvor f_e er positiv for alle kanter e

på vejen. Her kan vi forbedre ved at trække $\min\{f_e | e \in \mathcal{P}_2\}$ fra overalt på $\mathcal{P}_2$. Tilfældet med den modsat rettede vej er nok ikke det, man tænker først på, men der er jo ikke noget i vejen for, at sådanne veje findes i netværket, og man kan jo ikke udelukke, at den foregående chef for netværkstransporterne har bestemt, at der skulle transporteres noget den forkerte vej. Så kan vi straks score billige point på at aflyse denne transport i den forkerte retning (det vil iøvrigt vise sig, at strøm i bagudrettede kanter opstår på helt naturlig og rationel måde, så det drejer sig ikke om at tørre op efter andre).

I almindelighed vil vi nok ikke så ofte støde på disse rendyrkede tilfælde, men så kan vi prøve at se på den blandede udgave af de foregående: Antag, at der er en vej (der nu ikke længere er en orienteret vej, men blot kanter lagt i forlængelse af hinanden uden hensyn til orientering) $\mathcal{P}$ fra s til t, hvor der gælder

$$f_e < c_e$$

for de kanter, der er orienterede i retningen fra s mod t (da vejen starter i s og ender i t, kan vi jo altid afgøre om kantens rigtige orientering stemmer med vejens eller ej), og

$$f_e > 0$$

for de kanter, der er orienteret modsat, dvs. fra t mod s. Ved at bruge samme overvejelser som før kan vi konstruere en ny strøm f' med større værdi end f. Vi kan nemlig lægge

$$\min\{c_e - f_e | e \in \mathcal{P}, e \text{ orienteret fra } s \text{ til } t\}$$

til strømmen i hver kant, som er orienteret fra s til t, og trække

$$\min\{f_e | e \in \mathcal{P}, e \text{ orienteret fra } t \text{ til } s\}$$

fra strømmen i de øvrige kanter. Vi kalder en vej af denne type for *forbedrende* (mht. strømmen f). Vi har med andre ord, at såfremt der findes en forbedrende vej mht. strømmen f, da er f ikke maximal.

Faktisk har vi nu grundlaget for vor ønskede metode. Vi finder en forbedrende vej og laver den forbedring, som vejen giver mulighed for. Derefter finde vi en ny forbedrende vej, osv. Der mangler blot en sikkerhed for, at vi er færdige, når der ikke er nogen forbedrende vej.

Hertil bruges et begreb, som er særdeles vigtigt også til meget andet. Det drejer sig om et såkaldt *snit* (eng.: cut) i netværket, der stort set netop er, hvad det antyder at være, nemlig en mur lagt ned gennem netværket, så at s og nogle punkter er på den ene side, t og resten af punkterne på den anden. Formelt er et snit givet ved en klassedeling af V i to mængder W og U med $s \in W$ og $t \in U$, idet snittet er mængden af kanter $e = (v_1, v_2)$ med $v_1 \in W$ $v_2 \in U$. Det er praktisk at skrive

snittet som $\mathcal{F} = (W, U)$ (idet det underforstås, at $\mathcal{F}$ er alle kanter fra W til U). *Kapaciteten* af snittet $\mathcal{F}$ er

$$c(\mathcal{F}) = \sum_{e=(v_1,v_2)\in\mathcal{F}, v_1\in W} c_e.$$

Læg mærke til, at når vi udregner kapaciteten af et snit, så tager vi kun de kanter med, der går fra s's del af snittet og over til den anden del, ikke de kanter, der går den anden vej. Det kan se underligt usymmetrisk ud, men det er det ikke. Hvis vi tænker på snittet som om s breder sig ud og indlemmer alle punkter i sin snitmængde, så er kapaciteten naturligt defineret som den samlede godsmængde, som kan sendes ud af det nye "stor-s". At der eventuelt også kunne sendes noget ind, er uinteressant.

Betragt nu en strøm f for hvilken $f_e = c_e$ for alle e i et snit $\mathcal{F} = (W, U)$. Da en bevægelse fra s til t må passere en kant i snittet, og disse kanter allerede er fuldt belastede (man siger, at kanterne er "mættede"), tyder meget på, at strømmen er maximal. Her skal man imidlertid tage højde for orienteringerne; der kan gå strøm begge veje, og der er naturligvis ingen mening i at have en mættet strøm i den "forkerte" retning. Vi har dog følgende lille mellemresultat, som senere viser sig endog særdeles nyttigt:

Hvis f er en strøm, $\mathcal{F} = (W, U)$ et snit, da er

$$V(f) \leq c(\mathcal{F}).$$

Med andre ord, værdien af en vilkårlig strøm er altid mindre end kapaciteten af et vilkårligt snit. Tilsyneladende ikke noget at få åndenød over, men se lige, hvilken konsekvens det har: Hvis man for en given strøm kan finde et snit, for hvilket der gælder lighedstegn, så er strømmen maximal, for ingen strøm kan have større værdi en denne kapacitet!

Beviset for dette resultat går som følger: Vi starter med en helt ny måde at opgøre værdien af f på, nemlig

$$V(f) = \sum_{\substack{e\in\mathcal{F} \\ e\ s\text{-}t\text{-orienteret}}} f_e - \sum_{\substack{e\in\mathcal{F} \\ e\ t\text{-}s\text{-orienteret}}} f_e.$$

Hvis vi nemlig opgør strømmen ind og ud af hvert punkt i V og derefter lægger sammen, dels over W og dels over V, vil alle kanter internt i W eller U, altså alle pånær dem, der forbinder W og U, optræde to gange med hvert sit fortegn og dermed nette ud.

Benyttes dette i den foreliggende situation, har vi

$$V(f) \leq \sum_{\substack{e\in\mathcal{F} \\ e\ s\text{-}t\text{-orienteret}}} f_e = c(\mathcal{F}),$$

som giver den ønskede ulighed. □

4. Maximal strøm algoritmen

Hermed er den første algoritme for beregning af maximal strøm i et netværk klar til brug. Algoritmen går kort fortalt som følger:

Data: Et netværk $\mathcal{N} = (V, E, s, t, (c_e))$

Trin 1: Vælg en strøm f, f.eks. strømmen med $f_e = 0$ alle e og sæt initialværdien til $V(f)$

Trin 2: Undersøg om der findes en forbedrende vej. Hvis der gør det, ændres f til ny strøm med større værdi. Hvis der ikke findes en forbedrende vej, stoppes.

Resultat: Strømmen f.

Her er trin 2 en løkke i algoritmen, således at trinnet gentages, så længe betingelsen (der findes en forbedrende vej) er opfyldt.

Indtil videre har vi med algoritmen ovenfor kun givet en anvisning på en række beregninger, der skal gennemføres. Vi mangler at argumentere for, at algoritmen fører til det, vi ønsker, nemlig en maximal strøm. Vi har heller ikke umiddelbart nogen viden om, hvorvidt algoritmen overhovedet standser. Disse ting må vi se nærmere på.

Der startes med følgende:

Lad f være en strøm i et netværk $\mathcal{N}$. Hvis der ikke findes nogen forbedrende vej mht. f, da er f maximal, og værdien af f er lig med kapaciteten af et minimalt snit i $\mathcal{N}$.

Antag at f er maximal, og lad W være alle punkter v således at der findes en forbedrende vej fra s til p. Da f er maximal, ved vi at t ikke er i W. Vi betegner mængden $V \backslash W$ med U. $\mathcal{F} = (W, U)$ er et snit i netværket.

Vi har nu, at enhver kant e som begynder i W og ender i U er mættet, dvs. har $f_e = c_e$. Hvis det ikke var tilfældet, kunne vi nemlig finde en kant $e = (v_1, v_2)$ med $v_1 \in W$ og $v_2 \in U$ så $f_e < c_e$. Da der findes en forbedrende vej fra s til v_1 (sådan er W defineret), kan vi fortsætte den til en forbedrende vej fra s til v_2; det strider mod, at der ikke er nogen forbedrende vej fra s til et punkt i U.

Vi har også, at $f_e = 0$ for enhver kant, der går fra U til W. I modsast fald måtte der være en vej $e = (v_1, v_2)$ med $v_1 \in U$, $v_2 \in W$, så at $f_e > 0$. Da der findes en forbedrende vej fra s til v_2, ville denne kunne udvides til en forbedrende vej fra s til v_1, en modstrid.

Vi har, at $V(f)$ er summen af strømmene i kanterne fra W til U minus summen af strømmene fra U til W. Men de sidste har alle strømmen 0, og de første er nøjagtig

lig kapaciteterne. Med andre ord har vi, at

$$V(f) = c(\mathcal{F}).$$

Bruges nu resultatet fra forrige afsnit, ser vi, at f må være en maximal strøm og $\mathcal{F}$ et minimalt snit. □

Det er nu vist, at resultatet af algoritmen er det, vi leder efter. Det er også let at se, at algoritmen ikke kan vare ubegrænset længe, hvis alle kapaciteter er heltallige, for strømmens værdi øges med mindst 1 ved hvert gennemløb af trin 2, indtil den har nået sit maximum; det vil ske efter højest $c(\mathcal{F})$ gennemløb, hvor $\mathcal{F}$ er et minimalt snit.

Hovedresultatet af det foregående er sammenhængen mellem maximal strøm og minimalt snit. Det følger helt umiddelbart af vort sidste hjælperesultat.

Vi kan nu formulere konklusionerne af det foregående som følger:

Max-flow-min-cut sætningen: I ethvert netværk er værdien af en maximal strøm lig med kapaciteten af et minimalt snit.

Så vidt, så godt. Der mangler imidlertid noget i vor algoritme, for der henvises blot til, at der skal findes en forbedrende vej. Vi ved endnu ikke, hvordan man finder en sådan.

Hertil bruges en såkaldt *markeringsproces*. Ved at markere punkterne i netværket med par (v, r), hvor v er enten et punkt eller tallet 0, og r er et tal, eventuelt ∞, og ændre disse markeringer systematisk, kan vi søge frem til en forbedrende vej, såfremt der er nogen.

Søgeprocessen er som følger:

1. Start med at markere s med $(0, \infty)$.
2. Hvis $e = (v, w)$ er en umættet kant (hvor strømmen er mindre end kapaciteten) og v er markeret med (v', r), da kan w markeres med $(v, \min\{r, c_e - f_e\})$.
3. Hvis $e = (v, w)$ er en kant med $f_e > 0$, og w er markeret (v', r), da kan v markeres $(w, \min\{r, f_e\})$.
4. Der stoppes hvis der ikke er flere punkter at markere.

Man kan nu bruge markeringerne til at bestemme en forbedrende vej:

Mængden af markerede punkter svarer netop til de punkter v, for hvilke der findes en forbedrende vej fra s til v.

Hvis v er markeret, findes der pr. konstruktion en forbedrende vej fra s til v, idet markeringens første koordinat i hvert punkt angiver det punkt, som man skal gå tilbage til mod s. Tilsvarende vil anden koordinat af markeringen i v angive den værdiforøgelse af strømmen, som man kan få ved at bruge den forbedrendee vej.

Antag omvendt, at der findes en forbedrende vej fra s til v. Da vil ethvert punkt på denne vej kunne markeres, således at også v vil blive markeret. Der kan godt blive markeret andre punkter, men markeringsprocessen vil ikke standse, før alle punkter på vejen fra s til v er markeret. □

Algoritmen fra før skal dermed blot udbygges med markeringsprocessen, hvorved den samlede procedure bliver som følger: Der startes med en strøm 0. Derefter markeres punkterne; hvis markeringen inkluderer t, er der en forbedrende vej, og en sådan kan nu findes ved at gå baglæns ifølge anvisningerne fra markeringens første koordinat. Værdien af tilvæksten i strøm fremgår af anden koordinat i t's markering. Kanterne kan nu opdateres for den reviderede strøm, og der startes forfra med en ny markering.

5. Edmonds-Karp algoritmen

Proceduren udviklet ovenfor ender med det rigtige resultat, hvis den overhovedet ender. Det gør den, når alle kapaciteterne er heltallige (og i så fald er værdien af den maximale strøm iøvrigt også heltallig). Dette kan uden videre udvides til også at gælde, når alle kapaciteter er rationale tal, for dem kan vi jo gøre til hele tal ved at gange med deres fælles nævner.

Derimod er det mere tricky, hvis der er irrationale kapaciteter. Faktisk kan man komme ud for, at algoritmen slet ikke standser, og at de beregnede værdier ikke går mod den maximale strøm. I sig selv er forekomsten af irrationale kapaciteter noget, man kan tage med ro, for ved enhver beregning måtte man alligevel tilnærme dem med rationale. Det bagvedliggende problem er dog, at algoritmen i nogle situationer er meget langsom. Det kan der rådes bod på.

Den metode til at finde maximale strømme, som vi har betragtet i foregående afsnit, består af to trin, nemlig

(1) bestemmelse af en forbedrende vej,

(2) forbedring af den givne strøm på grundlag af den forbedrende vej.

For at undgå, at man i en vis forstand kører i ring, vil vi føje en detalje til, som sikrer kortest forbedrende vej; med denne tilføjelse kaldes proceduren for Edmonds-Karp algoritmen (Edmonds & Karp, 1972).

For at vi kan få gavn af at vælge korteste forbedrende vej, må vi have en metode til at sikre, at markeringsprocessen giver denne korteste vej. Det viser sig at være ret simpelt forudsat at der markeres på den rigtige måde:

Vi siger, at punktet v i netværket *behandles,* når samtlige punktet v', som kan markeres fra v, bliver markeret. Vi har da følgende resultat:

Hvis markeringen udføres således, at hvis v er markeret før v', da behandles v før v', så vil markeringen resultere i en korteste forbedrende vej.

Der startes med at behandle s; derved markeres $\mathcal{B}_1$ af punkter v i netværket

for hvilke længden af korteste forbedrendee vej fra s til v netop er 1.

I næste skridt behandles hvert af punkterne $v \in \mathcal{B}_1$, hvilket fører til markering af mængden $\mathcal{B}_2$ af punkter v' for hvilke længden af korteste forbedrendeforbedrende vej fra s til v' er 2. Dernæst behandles hvert af punkterne i $\mathcal{B}_2$, førende til markering af punkter i mængden $\mathcal{B}_3$, og så fremdeles, indtil punktet t markeres. □

Vi har hermed en algoritme til bestemmelse af maximal strøm i netværk. Hvordan man praktisk arrangerer sig med markeringsproceduren, ser vi på i næste afsnit.

En anden måde at betragte hele problemet (og en, der svarer mere til betragtningsmåden i kapitel 3) er denne: Antag at punkterne i netværket nummereres med $i = 1, 2, \ldots, m$, hvor s har nummeret 1 og t nummeret m, og at kanterne nummereres $j = 1, 2, \ldots, n$. Vi indfører også netværkets *incidensmatrix* $A = (a_{ij})_{i=1}^{m}{}_{j=1}^{n}$ givet ved

$$a_{ij} = \begin{cases} 1 & \text{hvis kant } j \text{ begynder i punkt } i \\ -1 & \text{hvis kant } j \text{ ender i punkt } i \\ 0 & \text{hvis kant } j \text{ ikke er incident med punkt } i \end{cases}$$

og vi kan nu skrive vort maximal-strøm problem som

$$\begin{aligned} &\max V \\ &\text{under bibetingelserne} \\ &A \begin{pmatrix} f_1 \\ f_2 \\ \vdots \\ f_n \end{pmatrix} = \begin{pmatrix} V \\ 0 \\ \vdots \\ -V \end{pmatrix} \\ &f_j \leq c_j,\ j = 1, \ldots, n \\ &f_j \geq 0 \end{aligned}$$

Her er første og sidste af ligningerne skrevet på matrixform betingelserne om at der skal gå en samlet strøm på V ud af s og ind i t; de øvrige betingelser siger, at der skal gå lige meget ind og ud af de øvrige punkter.

Vi ser, at maximal-strøm problemet faktisk lader sig formulere som et LP-problem (med variable $(f_1, \ldots, f_n)$ og V. Dermed er der også en løsningsmetode, nemlig f.eks. simplex-metoden for LP-problemer. Når vi i det foregående har betragtet en særlig algoritme, er det fordi den er hurtigere, idet den udnytter den særlige struktur ved LP-problemet ovenfor. Denne særlige struktur skyldes især matricen A, der dels har alle elementer fra mængden $\{-1, 0, 1\}$, dels har den ekstra egenskab, at summen af elementerne i enhver søjle er 0, fordi enhver kant starter i ét punkt, ender i et andet, og slet ikke er incident med de øvrige punkter.

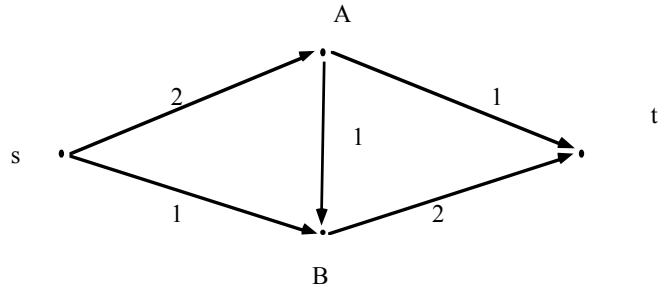
Det er også denne særlige struktur, der gør at LP-problemet ovenfor har en ekstra egenskab, som LP-problemer normalt ikke har, nemlig at hvis alle c_j er heltallige, da har problemet en heltallig løsning.

6. Markeringsprocessen

I gennemgangen af Edmonds-Karp algoritmen blev det sagt, at metoden til at finde forbedrende veje var en søgning i grafen, som ligger til grund for det givne netværk, idet man søger fra s ved "bredde-først". Det betyder, at man inddeler punkterne i afstandsklasser. Man markerer så først fra s til punkter i afstand 1, der nummereres i den orden, som de nås i. Denne nummerorden holder man så, når man går til den næste afstandsklasse, og så fremdeles.

Markeringen kan laves direkte i netværket angivet som graf, eller den kan laves når netværket repræsenteres ved en matrix.

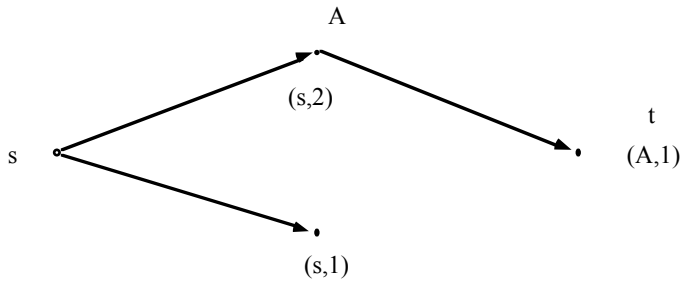
Eksempel 9.1



I dette forholdsvis banale netværk er det ikke svært umiddelbart at indsætte den maximale strøm, men det er ikke pointen. Vi ønsker her at angive systematikken i markeringsproceduren.

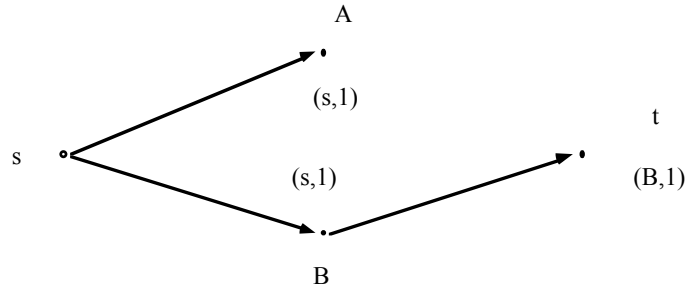
Vi starter med nulstrømmen og begynder at markere fra s. Herfra kan vi markere A med $(s, 2)$ (kanten kommer fra s og der kan sendes 2 igennem den). Tilsvarende kan vi markere B med $(s, 1)$ (bredde først!).

Så markeres der videre, først fra A, der var den først markerede i sidste rundte. Fra A kan vi markere t med $(A, 1)$ (der er en kant fra A, den har plads til 1, og vi har mindst 1 (nemlig 2) markeret i A). Nu er der en forbedrende vej, og der opdateres med 1 fra s over A til t.

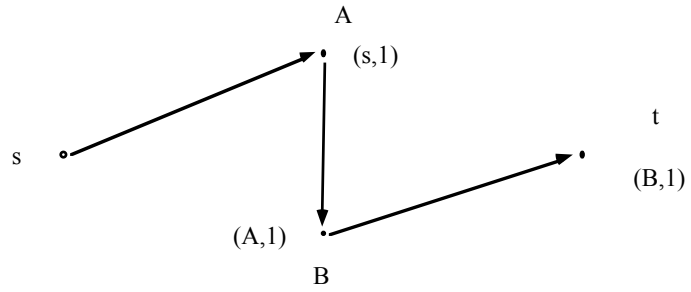


Vi starter på en ny markeringsrunde. Fra s kan vi igen markere til A, men nu kun med $(s, 1)$, og vi kan markere i B med $(s, 1)$. I næste skridt markerer vi fra A, men kanten til t er allerede mættet, så den kan vi ikke gå ad, og kanten til B er uinteressant, for der er jo allerede markeret i B. Så markerer vi fra B og der rammer vi t med $(B, 1)$ (der var 1 i B, og kanten kan kun have 2). Nu har vi en ny forbedrende vej fra s over B til t, og vi opdaterer med 1.

Eksempel 9.1, fortsat



Den tredie markeringsrunde starter med markering til A som tidligere. Vi kan nu ikke længere markere til B for kanten er mættet. Men så går vi til næste runde, og her kan der markeres i B med $(A, 1)$. I næste runde får vi markeret i t med $(B, 1)$ (der var 1 i B og kanten havde ledig kapacitet på 1). Der kan nu opdateres.



Så prøver vi fjerde markeringsrunde: Den kører tyndt fra starten, idet vi ikke kan markere til nogetsomhelst fra s. Dette fortæller os for det første, at vi har en maximal strøm (hvis værdi altså er 3). Men der kan trækkes lidt mere information ud: Alle de punkter, der kan nås fra s (i dette tilfælde altså s selv og ikke andre) giver os et minimalt snit (når vi tager komplementet som den anden af de to mængder, der definerer snittet).

Så simpelt behøver det ikke at gå. Vi skal nedenfor se et eksempel på et netværk, som i sig selv er simpelt nok, men hvor både markeringsproceduren og det minimale snit er mere komplicerede.

Alternativt kunne markeringerne være sket på tabelform. Først skriver vi netværket op som en tabel:

	A	B	t
s	2	1	
A	–	1	1
B		–	2

Markeringen kan ske ved at tilføje en ny linie til tabellen nedenfor. I denne linie skrives markeringerne på de respektive pladser, idet man starter i tabellen ovenfor ved linien fra s og søger at få fyldt den nye linie ud; når og hvis t-pladsen bliver fyldt ud, har man en forbedrende vej, og tabellen opdateres ved at der skrives strømme ind på de respektive pladser.

Første markering bliver derved repræsenteret ved linien

	$(s, 2)$	$(s, 1)$	$(A, 1)$

og der opdateres,hvorefter næste markering har formen

	$(s, 1)$	$(s, 1)$	$(B, 1)$

Eksempel 9.1, fortsat

Herefter får vi en markering på formen

	$(s,1)$	$(A,1)$	$(B,1)$

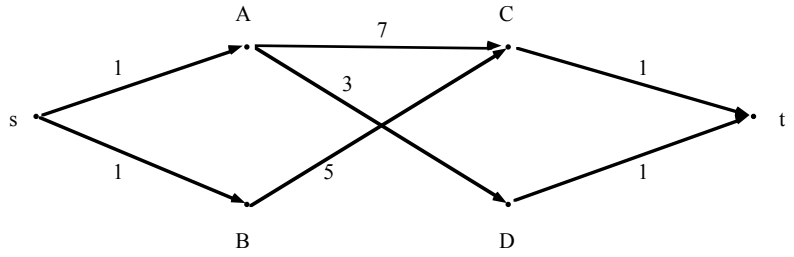
Som tidligere har vi nu, at et nyt forsøg på at markere går i vasken, og vi har en maximal strøm. Som nævnt har vi samtidig fundet et minimalt snit. Det er iøvrigt ikke så overraskende, at vi finder disse ting samtidig, det gør man jo normalt i lineær programmering.

Eksemplet var forholdsvis simpelt og tager ikke fuldt højde for alle de besværligheder, man kan komme ud for. Specielt kan man risikere at skulle markere ved at gå i modsat retning af en kant, hvor der i forvejen er strøm. Det ser vi et eksempel på senere; det foreliggende eksempel, som ihvertfald er let at overskue, fanger metoden i hovedtræk.

Vi har nu fundet ud af, hvorledes man finder forbedrende veje og opdaterer, når man i søgeprocessen ikke kommer ud for at bruge modsat orienterede kanter. Det kan man imidlertid godt risikere, og derfor må vi se lidt nærmere på denne komplikation.

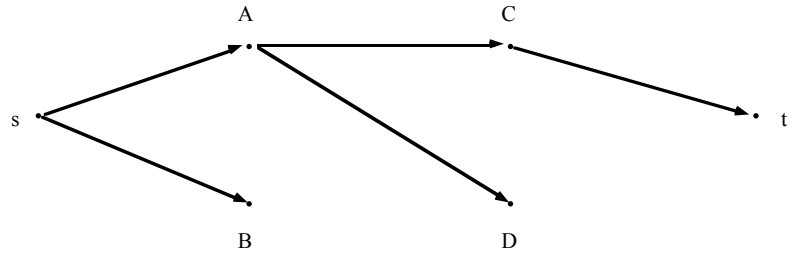
Eksempel 9.2

Betragt netværket givet nedenfor; det er igen ikke så svært umiddelbart at aflæse, at den maximale strøm er 2, men pointen er ikke den konkrete løsning, men metoden til at finde den. Vi kan jo sagtens komme ud for mere komplicerede netværk, hvor det ikke umiddelbart kan



aflæses.

Starter vi en sædvanlig løsningsproces, skal vi begynde med 0-strømmen og finde forbedrende veje ved markeringsproceduren. Den første forbedrende vej er der ikke noget besvær med, den finder vi ved at markere først fra s, dernæst videre fra A til C og D, og så fra C til t, hvormed den forbedrende vej er en realitet. Der kan bringes 1 igennem, og der opdateres i



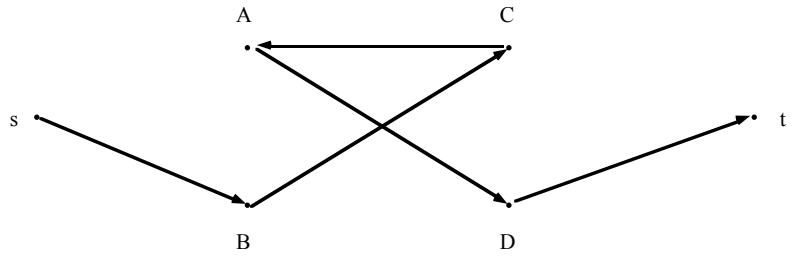
netværket.

Så går vi i gang med den næste markeringsrunde. Vi kan fra s kun markere til B, for kanten til A er mættet. Fra B kan vi kun komme til C, og fra C kan vi ikke komme videre til t, for denne kant er også mættet.

Tilsyneladende kan der ikke markeres igennem til t, og så skulle strømmen være optimal, men det ved vi jo fra en umiddelbar betragtning, at den ikke er. Altså må et eller andet være i vejen.

Eksempel 9.2, fortsat

Sagen er, at vi kan markere fra C til A mod kantens retning; der går nemlig en strøm 1 langs kanten fra A til C, vi kan bringe 1 frem til C, og så siger den generelle forskrift, at vi kan markere i A med $(C, 1)$. Fra A kan vi så komme videre til D, og fra D til t, hvormed vi har en forbedrende vej. Den er angivet neden-



for:

Der opdateres ved at vi tilføjer 1 i de kanter, hvor retningen er fra s til t og der reduceres med 1 (dvs. fra 1 til 0) i den kant, hvor strømmen er modsat. Det ses let, at vi nu har maximal strøm, for der kan simpelthen ikke markeres ud af s (det minimale snit er altså parret $(\{s\}, \{A, B, C, D, t\})$).

Intuitionen bag denne sidste forbedrende vej er, at vi blev nødt til at omdirigere strømmen fra A til C for at få ført en ekstra enhed frem. Omdirigering af en allerede indlagt strøm udføres teknisk ved bevægelser mod kantens orientering i den forbedrende vej. Man kan i det konkrete eksempel godt se, at vi havde undgået denne omdirigering, hvis vi f.eks. systematisk havde startet markeringerne nedefra (dvs. B før A), men det er jo en efterrationalisering; når vi står ved starten af algoritmen og har et tilpas kompliceret netværk, kan man ikke gennemskue tingene på den måde, og derfor er man nødt til at holde muligheden for at gå mod orienteringen åben.

Det kan med nogen ret hævdes, at det er besværligt at holde styr på baglæns bevægelser af den type, vi har set ovenfor. Det kan måske endda gå, når netværket er givet ved sin plane repræsentation (eller, sagt med jævne ord, tegnet på et stykke papir som i eksemplet ovenfor), men hvis det er given ved en matrix, så er det temmelig overskueligt (selvom det godt kan lade sig gøre, når man holder tungen lige i munden). Det er væsentlig mere ligetil at markere i kanternes retning, så derfor vil vi angive en metode, der udelukkende benytter markering fremad.

Da man ikke kan undvære baglæns markering, må proceduren gå ud på at erstatte netværket af et andet (men selvfølgelig et, som er så tæt knyttet til det oprindelige, at der ikke går noget galt ved det). Vi kalder dette hjælpenetværk for *residualgrafen;* ved start er det lig med netværket selv, men det opdateres systematisk efter hver forbedrende vej. Systematikken er følgende: Hvis den forbedrende vej har en vis strøm i en kant, så gør man to ting:

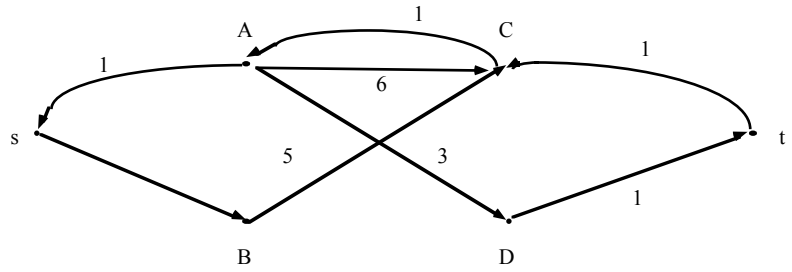
(i) kapaciteten reduceres med denne strøm (hvis kapaciteten derved bliver 0, fjernes kanten), og

(ii) der tilføjes modsat rettet kant med strømmen som kapacitet (og hvis der var sådan en i forvejen, så øges kapaciteten med strømmens størrelse).

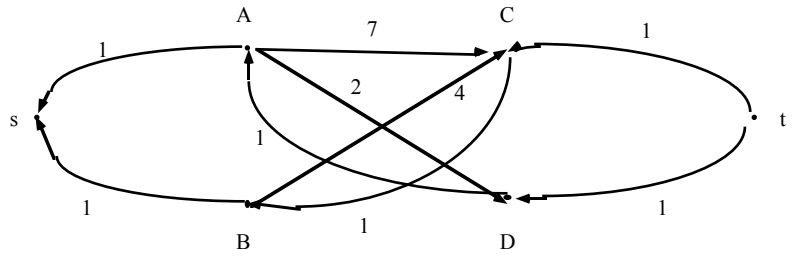
Den næste forbedrende vej findes nu i residualgrafen, og denne opdateres, og så fremdeles.

Eksempel 9.3

Lad os se eksemplet ovenfor lavet med residualgraf. Ved starten er det netværket selv. Så finder vi den forbedrende vej ligesom før. Men nu opdateres der i overensstemmelse med forskriften ovenfor. Den nye residualgraf har vi her:



De buede kanter er tilføjede, og kapaciteterne er sat på i overensstemmelse med forskriften. Der markeres nu (kun fremad!) i residualgrafen; det er let at se, at man får samme forbedrende vej som før. Ved opdateringen forsvinder den kunstige kant fra C til A (kapaciteten reduceres jo til 0) og der tilføjes 1 til kapaciteten på den modsat rettede kant, så den kommer tilbage til de oprindelige 7. Tilsvarende ved de øvrige kanter. Ialt fås den nye residualgraf som vist nedenfor. Det er igen let at se, at der ikke kan laves flere forbedrende veje; man kan slet ikke komme ud af s.



Det er måske ikke oplagt, at denne metode er mere bekvem til at holde øje med omvent orienterede kanter end den direkte metode. Fordelen kommer faktisk klarest frem, når man betragter markeringsprocessen på tabelform. Så det gør vi.

Netværket fra før kan skrives op som tabel således som vist nedenfor; det er samtidig residualgrafen i første markering.

	s	A	B	C	D	t
s		1	1			
A				7	3	
B				5		
C						1
D						1
t						

Første markering er som tidligere, for når man starter med nulstrømmen, kan det alligevel aldrig komme på tale at markere mod orienteringen. Markeringen bliver dermed til:

s	A	B	C	D	t
	$(s,1)$	$(s,1)$	$(A,1)$	$(A,1)$	$(C,1)$

Der skal så opdateres, og det sker som tidligere ved at indføre den faktiske strøm i den rubrik, som svarer til en kant, øverst til venstre.

Eksempel 9.3, fortsat

Men vi gør mere end det: Vi tager den symmetrisk beliggende kant, og i den indføjer vi strømmen øverst til *højre*. F.eks. skal vi i C-række og A-søjle skrive et 1 øverst til højre (og tilsvarende i A-række, s-søjle, og i t-række, C-søjle). Hvis der i forvejen stod noget, ændres det op eller ned med den nye opdatering.

Vor tabel kommer herefter til at se således ud:

	s	A	B	C	D	t
s		1 1	1			
A	1			1 6	3	
B				5		
C		1				1 1
D						1
t				1		

Vi kan nu starte forfra med at markere i den opdaterede tabel, idet vi regner tallene i øverste højre hjørne (= kapaciteter på nye kunstige kanter) med. Der startes med at markere fra s til B, idet der ikke kan markeres langs kanten til A, som er mættet. Så går vi videre: A-rækken kan ikke bruges, for vi fik jo ikke ført noget frem til A fra s. Så går vi til B-rækken, hvor der kan markeres i C. Dernæst går vi videre: i C-rækken kan vi markere til A langs den nye kunstige kant, men så heller ikke andet. Vi kan ikke bruge D-rækken, da D ikke er markeret endnu, og heller ikke t-rækken. Så vender vi bøtten og starter forfra med s-rækken. Den giver ikke noget nyt, men når vi derefter går til A-rækken, kan vi markere i D. Vi fortsætter ned over rækkerne, der ikke giver noget, før vi kommer til D, men så er den der til gengæld også. Ialt ser vore markeringer således ud:

s	A	B	C	D	t
	$(C,1)$	$(s,1)$	$(B,1)$	$(A,1)$	$(D,1)$

Fra denne markering kan vi genskabe den forbedrende vej, idet vi starter bagfra: Vi nåede t fra D. Et blik i D-søjlen fortæller os, at vi kom dertil fra A, og i A-søjlen står, at vi kom til A fra C. C nåede vi fra B, hvortil vi kom fra s. Der står også at læse, at der kan sendes 1 igennem. Tabellen skal derfor opdateres med 1, hvad der fører til nye posteringer i en række felter ifølge reglerne ovenfor. I feltet svarende til en kant fra C til A kommer der en strøm på 1, hvilket skrives ved at nedjustere tallet i højre hjørne tilsvarende, dvs. til 0, og så kan vi lige så godt lade være at skrive det. Ialt får vi derfor følgende tabel

	s	A	B	C	D	t
s		1 1	1 1			
A	1			7	1 2	
B	1			1 4		
C			1			1 1
D		1				1 1
t				1	1	

Eksempel 9.3, fortsat

Næste markeringsrunde går i stå allerede fra starten; der kan ikke markeres ud af s, og dermed er der ikke noget at markere videre fra. Som altid kan vi finde s-mængden i det optimale snit ved til s at føje alle de punkter, der kan markeres. Dem var der ingen af, så vi ser (igen), at snittet er $(\{s\}, \{A, B, C, D, t\})$.

7. Dinic's algoritme

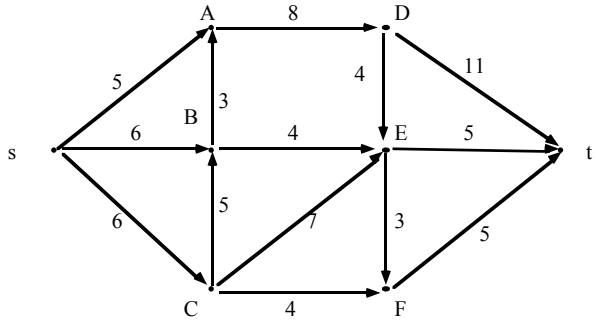
Det blev nævnt i gennemgangen af Edmonds-Karp algoritmen, at den særlige forskrift om rækkefølgen af afsøgning i markeringsproceduren sikrer. at algoritmen finder en maximal strøm i løbet af et antal beregningsskridt, som ikke overstiger m^5, hvor m er antallet af punkter i netværket. Det lyder måske af meget, men set i lyset af, hvor meget arbejde der er med f.eks. generelle heltalsproblemer, så er det faktisk et tegn på, at vi har at gøre med en ret veltrimmet algoritme.

Ikke desto mindre kan man jo altid ønske, at tingene forløber noget hurtigere, og det kan man faktisk også opnå. Der findes flere forbedringer af Edmonds-Karp algoritmen, og den, som vi vil kigge nærmere på, benytter stort set samme metode. Det drejer sig om Dinic's algoritme, som sikrer løsning i højest m^4 skridt, idet man kan slå en række forbedrende veje sammen i én fælles forbedring.

Metoden gør brug af afstandsklasser, som vi kiggede lidt på i kapitel 6. Ved at holde øje med afstanden fra s, når vi markerer, kan vi ved én og samme markering fange alle forbedrende veje af en vis længde. Vi går systematisk frem og finder alle de forbedringer, som har kortest mulig afstand. Dermed stiger afstanden hele tiden, og da der højest kan være afstanden m, har vi en grænse på antal skridt. Eneste ulempe er, at vi skal være lidt mere omhyggelige for at fange samtlige forbedrende veje af en given længde ved en og samme markering.

Eksempel 9.4

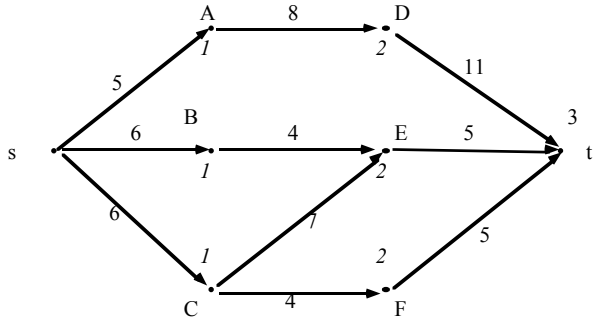
Vi viser Dinic's algoritme ved et eksempel:



Vi starter som sædvanlig med nulstrømmen i netværket. Derefter indlægger vi afstandsklasser, hvilket kan gøres ved at markere stort set på samme måde som tidligere, blot behøver vi ikke holde styr på andet end afstanden fra s.

Eksempel 9.4, fortsat

I første omgang markeres A, B og C med 1; derefter kan vi markere videre fra A til D, som får 2, og tilsvarende fra B til E og fra C til F. Endelig kan t nu markeres med 3. Hvis der, når vi kommer til t, er punkter, som ikke er markeret, ser vi bort fra dem i det følgende, ligesom kanter, der går mellem andet end to efterfølgende afstandsklasser, negligeres. Det resulterende netværk er følgende:

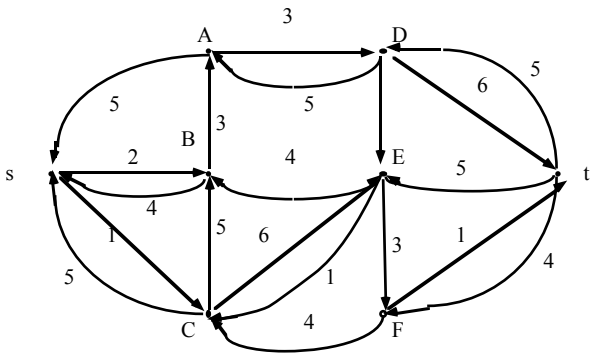


Nu skal vi have fundet en maximal strøm bestående af forbedrende veje med afstand 3. Det gør vi på følgende måde: Først noterer vi for hvert punkt den maximale *gennemløb,* og det finder vi på følgende måde: Vi tager alle kanter mellem punktet og den foregående afstandsklasse, og for hver finder vi *nyttekapaciteten* som forskel mellem kapacitet og faktisk strøm, hvis kanten peger ind i punktet, og faktisk strøm hvis den peger ud. Summen af disse er kantens *inputkapacitet;* tilsvarende finder vi *outputkapaciteten,* idet vi nu blot bruger kanter mellem punktet og den følgende afstandsklasse; nu er gennemløbet det mindste af input- og outputkapaciteterne.

Når vi har fundet gennemløb for alle punkter, kan vi gå igang med at sætte strøm på. Vi vælger et punkt med minimalt gennemløb. Så finder vi en forbedrende vej fra punktet til t og fra s til punktet. Der opdateres med den strøm, som nu er tildelt, og proceduren gentages indtil netværket er blevet usammenhængende.

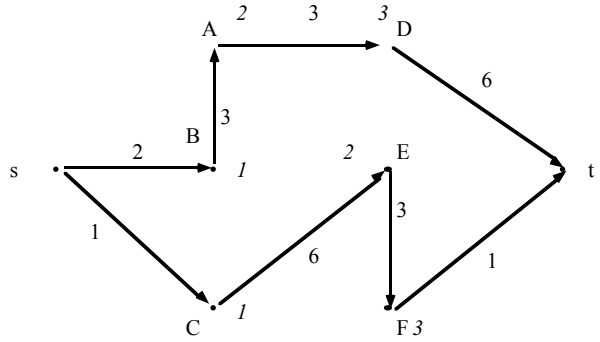
I eksemplet ser vi, at det minimale gennemløb er 4, som realiseres i punkterne B og F. Vi vælger B og sender 4 til t via E; klart nok skal de fire komme fra s. Vi kan så fjerne punktet B og kanterne til og fra B (generelt fjernes alle punkter med gennemløb 0 og de kanter, der fører til og fra sådanne), og vi kan gå videre. Nu er gennemløbet i E det mindste, nemlig 1 (inputkapaciteten er 7, outputkapaciteten 1, idet de oprindelige 5 er reduceret med strømmen 4). Vi sætter strøm 1 fra E til t og lader den komme fra s over C til E. Så går vi videre til puntet F som nu har lavest gennemløb 4, hvilket giver strøm 4 fra s over C over F til t, og endelig skal der en strøm 5 gennem A, nemlig fra s og via D til t. Dermed er der slet ingen punkter tilbage mellem s og t så netværket er klart nok blevet usammenhængende.

Vi er nu færdig med første runde og skal opdatere vort oprindelige netværk. Strømmene har vi fundet undervejs, men vi skal også lave en ny residualgraf, der får følgende form:



Eksempel 9.4, fortsat

Fra denne kan vi først finde afstandsklasser og kanter imellem disse, og det giver os følgende netværk:



Her er det let at se, hvorledes der kan laves forbedring, men det sikreste er jo at følge den standardiserede fremgangsmåde. Det betyder, at vi skal vælge C med laveste gennemløb 1; derved får vi strøm 1 fra s over C, E og F til t, og den nedre del forsvinder. Det nye minimum 3 nås i A, og det giver strøm 3 på oplagt måde.

Dinic's metode benytter, som vi så, ikke forbedrende veje, men forbedrer hele delnetværk ad gangen. Disse er konstruerede sådan, at der kun er kanter mellem en afstandsklasse og dens umiddelbare forgænger og efterfølger. Det er denne egenskab, der sikrer, at man kan finde maximal strøm ved at opsøge punkter med minimalt gennemløb; det går ikke for et vilkårligt netværk.

8. Parring i grafer

Som et biprodukt af vore overvejelser omkring maximal-strøm algoritmen fik vi den berømte max-flow-min-cut sætning; den har vi endnu ikke rigtig brugt til noget, og det kan give en forkert opfattelse af dens betydning, for den har mange anvendelser. Egentlig er det en udgave af de generelle dualitetsresultater for optimeringsproblemer, med alt hvad deraf følger.

Vi skal se på et par anvendelser, der ikke er umiddelbart operationsanalytiske, men som på sin side finder anvendelse i andre, og mere praktisk orienterede problemer.

Det første har at gøre med parring (eng.: matching) i grafer. Givet en graf $\Gamma = (V, E)$; en *parring* i Γ er en mængde $\mathcal{M}$ af kanter i grafen, således at ethvert punkt i V er incident med højest én kant i $\mathcal{M}$. Endepunkterne af en kant i $\mathcal{M}$ kaldes parrede. Vi siger, at $\mathcal{M}$ er *maximal* hvis der ikke er nogen parring med større antal kanter end M, og *perfekt* hvis alle punkter i V er parrede.

En *overdækning* i en graf er en mængde $\mathcal{C}$ af punkter i V således at enhver kant i E er incident med mindst ét punkt i $\mathcal{C}$. Vi har da følgende resultat:

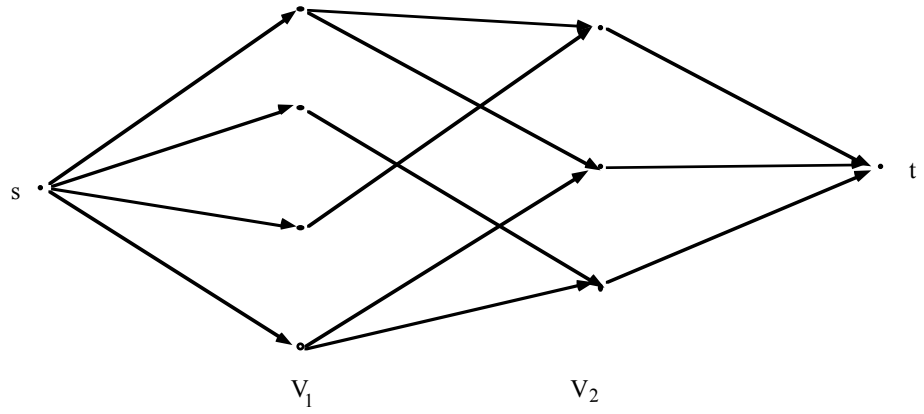
Lad $\mathcal{M}$ være en parring i Γ og $\mathcal{C}$ en overdækning. Da gælder

$$|\mathcal{M}| \leq |\mathcal{C}|.$$

For hvert $e \in \mathcal{M}$ må mindst et af endepunkterne tilhøre $\mathcal{C}$. Da ingen af disse endepunkter kan optræde mere end en gang, når e gennemløber $\mathcal{M}$, har vi resultatet. □

Det generelle problem om at finde maximale eller perfekte parringer er temmelig kompliceret; her nøjes vi at se på parringer i *todelte* grafer, hvor punktmængden V kan opdeles i V_1 og V_2 således at enhver kant forbinder et punkt i V_1 med et i V_2.

Til en todelt graf Γ vil vi knytte et netværk $\mathcal{N}$ på følgende måde: Vi orienterer hver kant i Γ ved at lade dem gå fra V_1 til V_2. De får kapacitet $+\infty$ (eller et tilpas stort tal K). Vi tilføjer et nyt punkt s og en kant (s, v_1) for hvert punkt i V_1, samt et nyt punkt t og en kant (v_2, t) for hvert punkt $v_2 \in V_2$. Disse sidste kanter får alle kapaciteten 1. Denne konstruktion er vist i figuren.



Lad $\mathcal{M}$ være en parring i Γ. Svarende til $\mathcal{M}$ kan vi lave en strøm f i det konstruerede netværk ved at sætte strømmen i alle kanter fra $\mathcal{M}$ til 1, og tilsvarende ved at lade strømmen i alle kanter fra s til en punkt i V_1 incident med en kant fra $\mathcal{M}$ være 1, samt endelig at sætte strømmen til 1 for alle kanter fra et punkt i V_2 incident med en kant fra $\mathcal{M}$ til t. Alle øvrige kanter får strømmen 0. Vi får, at $V(f)$ er et heltal; det er iøvrigt lig med antallet af kanter i $\mathcal{M}$. Omvendt vil enhver strøm i netværket, for hvilket $V(f)$ er heltallig, kunne identificeres med en parring i grafen bestående af ialt $V(f)$ kanter.

Konklusionen heraf er, at man kan finde en maximal parring i en todelt graf ved at finde en maximal strøm i $\mathcal{N}$. Med andre ord, man kan bruge Edmonds-Karp algoritmen (eller en modifikation) heraf, til at finde maximale parringer i todelte grafer. Men man kan mere endnu; max-flow-min-cut sætningen giver os en forbedring af resultatet for todelte grafer, den såkaldte *König's sætning:*

König's sætning: Lad Γ være en todelt graf, $\mathcal{M}$ en maximal parring i Γ, og $\mathcal{C}$ en minimal overdækning i Γ. Da gælder

$$|\mathcal{M}| = |\mathcal{C}|.$$

Vi har fra hjælperesultatet ovenfor, at der altid gælder $|\mathcal{M}| \leq |\mathcal{C}|$. Lad $\mathcal{F} = (W, U)$ være et minimalt snit i netværket $\mathcal{N}$ hørende til grafen. Vi ved, at vort netværk har et snit med kapaciteten $|V_1|$ (nemlig snittet $(\{s\}, V_1 \cup V_2 \cup \{t\})$, så hvis tallet K er valgt større end $|V_1|$, kan det minimale snit ikke indeholde kanter fra V_1 til V_2 (de har nemlig hver kapacitet K og bare én af dem giver derfor en større samlet kapacitet). Alle kanter fra W til U er altså enten en kant fra s til V_1 eller en kant fra V_2 til t.

Tag nu skæringsmængderne $W' = W \cap V_2$ og $U' = U \cap V_1$. Vi påstår, at enhver kant fra V_1 til V_2 må røre enten W' eller U'. I modsat fald måtte der være en kant fra $V_1 \backslash W$ til $V_2 \backslash U$, f.eks. fra $v_1 \in V_1$ til $v_2 \in V_2$. Fra definitionen af det minimale snit som de punkter, der kan nås ved en forbedrende vej fra s får vi, at v_2 kan nås ved forbedrende vej, og den må derfor ligge i W, en modstrid.

Så er vi ved at være der: Da alle kanter rører ved enten W' eller U', er $W' \cup U'$ en overdækning. Dens størrelse kan måles som antal punkter i W' = antal kanter fra s til W' plus antal punkter i U' = antal kanter fra U' til t, altså som kapaciteten af det minimale snit. Men den er jo lig med den maximale strøm, det vil sige det maximale antal elementer i en parring. □

Eksempel 9.5

Det kan lette forståelsen af argumenterne bag König's sætning, hvis man bruger den særlige fortolkning af parring og overdækning, som allerede er latent i betegnelserne:

Antag at vi har en befolkning V opdelt i hun- og hankønsvæsener (henholdsvis V_1 og V_2). At der er en kant mellem punkter fra de to mængder, vil vi fortolke som at de to personer kommer sammen. Der ses bort fra at personer af samme køn kan pleje omgang; vi har at gøre med en todelt graf.

Vi ønsker nu at finde ud af, hvor mange ægteskaber, der kan komme ud af den foreliggende situation, idet bindingen her er den oplagte, at bigami er forbudt; hver person må kun deltage i ét ægteskab. Det er oplagt, at dette svarer til en parring, og vi er nu interesserede i at få så mange ægteskaber som muligt ud af de eksisterende bekendtskaber.

En overdækning får vi, hvis vi ønsker en liste af personer, som skal have tilsendt en AIDS-pjece, og listen skal være så stor, at mindst én deltager i ethvert parforhold, ægteskabeligt eller ej, skal have fået pjecen. Hvorledes finder vi frem til det mindst mulige antal pjecer?

Her er et forslag (der følger slagets gang i König's sætning). Vi allierer os med EU's informationskontor og bestiller værket "Familien i det nye Europa", der består af dels en række artikler om EUs familiepolitik, dels en kommenteret gennemgang af samtlige EU-dokumenter om harmoniseringen af det ægteskabelige og uægteskabelige samliv i det europæiske fællesskab. Denne bog udleveres i ét eksemplar, med personlig dedikation skrevet af den relevante kommissær, til alle af hunkøn, og samtidig annonceres det, at alle af hankøn har ret til at aflevere netop ét eksemplar af denne bog på kommunernes affaldsopsamlingspladser.

Eksempel 9.5, fortsat

Der er – ganske rigtigt – regnet med, at alle, som ligger inde med denne bog, benytter første lejlighed til at give den videre. Kvinder, der ikke kan aflevere den til nogen, må til krisebehandling. Hvis manden, som får den, kun modtager denne ene, afleverer han den til kommunen og er ude af problemet. Hvis han får flere, vil han forsøge at levere tilbage til dem, de kom fra, og undervejs i denne proces må også han søge professionel hjælp hos krisebehandlerne. Hvis bogen vender tilbage, vil kvinden forsøge at dumpe den hos eventuelle andre bekendte, der så enten kan skaffe sig af med den eller må gå til krisebehandling.

Vi går ud fra, at denne proces nu er løbet til ende. Lad os så lave en liste bestående af

(1) kvinder, hvis bog er blevet afleveret videre og endt tilbage hos kommunen,

(2) mænd, som har søgt krisebehandling.

Hvis vi tager et eller andet parforhold, så må enten kvinden høre til gruppe (1) eller manden til gruppe (2), for ellers var der en kvinde med en bog, der kom sammen med en mand, der endnu ikke havde afleveret nogen, og det ville give en mulighed for endnu en aflevering. Men det betyder lige akkurat, at alle parforhold representeres ved vor liste, som derfor er en kandidat til adresseliste for AIDS-pjecen.

Hvor mange pjecer skal sendes ud? Her bruger vi max-flow-min-cut på netværket lavet ved at tilføje kilden s, der er udleveringsstedet for EU-bøger, og t, der er afleveringsstedet. Kapaciteten af det (minimale) snit, vi lavede, kan opgøres som summen af de bøger, der blev afleveret til kvinderne i (1), og de bøger, der blev afleveret af mændene i (2). Det svarer til vor adresseliste; da snittet var minimalt, er det samtidig værdien af den maximale strøm.

Men den maximale strøm er netop det størst mulige antal ægteskaber: Det kan passende baseres på de fælles erindringer om den pressede situation, man var i, da man skulle have det digre EU-værk ekspederet ud. På grund af reglerne om kun én aflevering pr. mand kommer der ikke bigami ud af det.

Ialt har vi dermed en adresseliste bestående af lige så mange adresser, som der er ægteskaber. Denne liste er dermed minimal, for man kan selvfølgelig ikke have færre adresser end ægteskaber, hvis samtlige forbindelser skal være dækket. Vi har således at antal kanter i en maximal parring er lig med antallet af punkter i en minimal overdækning.

Hele proceduren kan eventuelt udbygges med fuld TV-dækning som et populært underholdningsprogram op til de regelmæssigt tilbagevendende EU-valg.

Det er ikke sikkert, at en todelt graf har en parring som omfatter alle punkter, en såkaldt *perfekt* parring. Men König's sætning kan bruges at give en betingelse for, at der findes en sådan parring:

Lad Γ være en todelt graf med $V = V_1 \cup V_2$. Da har Γ en perfekt parring netop hvis der for enhver delmængde V' af V_1 gælder at der er mindst $|V'|$ punkter i V_2, som er forbundet med en kant fra Γ til et punkt i V'.

Hvis Γ har en perfekt parring, er det oplagt, at ethvert punkt i en hvilkensomhelst delmængde V' er forbundet til et punkt i V_2, og dette sidste punkt optræder kun en gang, så konklusionen følger umiddelbart.

Omvendt: Antag at hver delmængde V' af V_1 er forbundet ved en kant med mindst $|V'|$ punkter i V_2. Vælg en minimal overdækning $\mathcal{C}$ i Γ og sæt mængden af punkter i henholdsvis $V_1 \cap \mathcal{C}$ og $V_2 \cap \mathcal{C}$ til henholdsvis t_1 og t_2. Bruger vi nu betingelsen på de V_1-medlemmer, som *ikke* er i overdækningen, får vi for det første, at der er $|V_1| - t_1$ af dem; for det andet har vi, at de kun kan være forbundne til de

t_2 punkter i $V_2 \cap \mathcal{C}$ (ellers ville overdækningen slet ikke røre den pågældende kant, en modstrid. De $|V_1| - t_1$ punkter er altså kun forbundne til de t_2, men så kan vi fra den generelle betingelse aflæse at

$$|V_1| - t_1 \leq t_2$$

eller

$$|V_1| \leq t_1 + t_2.$$

Da $t_1 + t_2 = |\mathcal{C}|$ og V_1 også er en overdækning, er $|V_1| = |\mathcal{C}|$. Af König's sætning fås nu, at der findes en parring med V_1 kanter. □

Eksempel 9.6

Man kan også servere Hall's ægteskabs-sætning i den lidt mere jordnære udgave brugt i det foregående eksempel (det havde også været mærkeligt andet). Lad os om et samfund antage, at enhver pigeklub er i stand til at arrangere et party af deres bekendtskaber, så hver pige har mindst en fyr til egen disposition. Vi skal da vise, at alle piger kan blive gift.

Her bruger vi den allerede fundne adresseliste for AIDS-pjecer. Hvis vi laver en klub bestående af piger, som ikke har fået noget pjece, og sætter dem til at arrangere et party; de inviterede fyre skal tage deres pjece med, så man kan studerede den sammen. Det kan lade sig gøre: fra vor antagelse ved vi, at pigerne har bekendtskaber nok, at og da adresselisten var baseret på en overdækning (i ethvert bekendtskab kender den ene af dem pjecen), må de fyre, som pigerne kender, have fået pjecen.

Vi kan nu slutte, at der er udsendt mindst lige så mange AIDS-pjecer, som der er piger: Enten har pigerne selv fået den, og hvis ikke, så kan de pågældende finde hver sin fyr, der har fået den. Så bruger vi König: Da vor adresseliste var en minimal overdækning, er den lig med antallet af elementer i en maximal parring, og det vil her sige et maximalt antal ægteskaber. Der kan altså laves mindst lige så mange ægteskaber, som der er piger.

9. Udvidelser af max-flow-min-cut

Vi har i det foregående hele tiden antaget, at der var kapacitetsgrænser opad på strømmen langs de enkelte kanter i netværket, men at den eneste begrænsning nedad var, at strømmen ikke måtte være negativ. Det kan man godt udvide en smule: Det er muligt at indføre *nedre begrænsninger* på strømmene, dvs. tal $l_{ij} \geq 0$, fortolket således, at der skal passere mindst l_{ij} langs kantet fra i til j.

Denne ekstra antagelse ændrer ikke ved grundprincippet, men den kræver dog lidt tilpasning. For det første kan vi ikke længere bare starte med nulstrømmen, for den er ikke nødvendigvis mulig (den overtræder nu de nedre begrænsninger). Men har vi først en mulig strøm, kan vi som tidligere tale om forbedrende veje og vise, at vi har en maximal strøm, når der ikke er mulighed for forbedring. Der skal rettes lidt på kriterierne for en forbedrende vej (hvis kanten er orienteret modsat og der går strøm igennem, skal denne strøm være større end den nedre begrænsning, og det er kun denne del, der tæller), men disse rettelser er ret indlysende.

Det samme gælder egentlig også for den teoretiske del af historien, nemlig sætningen om at maximal strøm er lig med minimalt snit. Vi skal vare passe på at definere kapaciteten af snittet $\mathcal{F} = (W, U)$ på rette måde, nemlig som

$$c(\mathcal{F}) = \sum_{i \in W, j \in U} c_{ij} - \sum_{i \in U, j \in W} l_{ij}.$$

Det er sidste led, som er det nye; egentlig nyt er det dog ikke, for da de nedre grænser var 0 i det foregående, var denne sum altid 0, og så behøvede vi ikke at skrive den, selvom den egentlig var der.

At der stadig gælder max-flow-min-cut, er ikke så svært at indse, og det kan vi godt overlade til eget initiativ. Man kan gennemføre det som i den tidligere udgave, eller man kan også lave et trick med at trække den aktuelle strøm fra kapaciteter på alle kanter, se på det fremkomne (gammeldags) netværk, bruge max-flow-min-cut her, og så vende tilbage til det oprindelige.

Det kan også være nyttigt at vide, at man kan minimere i stedet for at maximere. I første omgang noterer vi os, at der jo ikke ændres noget ved at der til alle data i netværket lægges et positivt tal, eller at der trækkes et positivt tal fra. Med andre ord kan man godt løse netværksproblemer, hvor alle kapaciteter (og dermed alle strømme) er negative tal. En maximal negativ strøm er så en, der har minimal numerisk værdi.

Men så kan vi også løse minimeringsproblemer af typen, hvor der er givet en nedre begrænsning c_{ij} og en kapacitetsgrænse l_{ij} for hver kant (i, j) (det er med vilje, at notationen tilsyneladende er byttet om). Vi tager nemlig og maximerer $-\sum_i x_{(s,i)}$ (det der går ud af s) for netværket med kapaciteter $-c_{ij}$ og nedre grænser $-l_{ij}$. Dette er et sædvanligt maximeringsproblem, som vi kan løse, hvorved vi faktisk finder den minimale strøm i det oprindelige problem. Vi ved endda også, at max-flow-min-cut holder for dette problem, og det oversætter til, at værdien af den minimale strøm bliver lig med maximum af

$$\sum_{i \in W, j \in U} l_{ij} - \sum_{i \in U, j \in W} c_{i,j}$$

over alle snit (W, U) i det oprindelige netværk. Dette er den oplagte kopi af max-flow-min-cut, nemlig min-flow-max-cut-sætningen.

Der er en tredie grafteoretisk klassiker (foruden König's og Hall's sætninger), som kan bevises ved hjælp af min-flow-max-cut: Det drejer sig om *Dilworth's sætning,* der f.eks. kan formuleres på følgende måde:

Dilworth's sætning: Lad G være en orienteret graf uden orienterede cykler og lad A være en delmængde af dens kanter. Det mindste antal orienterede veje som er nødvendige for at dække alle kanterne i A er lig det maximale antal kanter i A, således at intet par kan udvides til en orienteret vej.

Tilføj (som sædvanlig) punkter s og t til grafen, og lav kanter (s,i), (i,t) for alle punkter i bortset fra s og t. Tilføj nedre grænser $l_{ij} = 1$ for kanterne i (i,j) i A og $l_{i,j} = 0$ for de nye kanter. En minimal strøm giver dermed det mindste antal veje, der bruger alle kanter i A. Denne strøm er lig den maximum af

$$\sum_{i \in W, j \in U} l_{ij} - \sum_{i \in T, j \in S} c_{ij}$$

over alle snit (W,U). For at få denne sum stor skal man have så mange af A-kanterne med som mulig ved overgang fra S til T; hvis man imidlertid får to A-kanter med, der kan forbindes med en orienteret vej, er der en af vejens kanter (ikke nødvendigvis fra A), der passerer snittet i gal retning (fra U til W) og så trækkes der $+\infty$ fra i summen ovenfor. Derfor vil alle de A-kanter, der tæller med, være sådan, at intet par kan indgå i en vej. □

Man kan umiddelbart synes, at Dilworth's sætning (der er fra 1950) lyder lidt vel teoretisk til at være anvendelig, men det skal man ikke lade sig narre af, for den har faktisk ret mange anvendelser.

Eksempel 9.6

Hvordan man undgår overflødige Airbusser. Den prominente anvendelse er allokering af maskiner på ruter i et flyselskab. Vi kan tænke os rutenettet som en orienteret graf, som formodentlig alle udgår fra et hjemsted. Hver af ruterne (kanterne) flyves igennem en gang den ene vej og en gang den anden, og orienteringen angiver den første gennemflyvnings retning. En maskine kan tage flere ruter efter hinanden, hvis det ellers kan lade sig gøre. Hvor mange maskiner skal bruges til at betjene nettet, når man ikke ønsker at forkøbe sig på overflødige maskiner?

Her fortæller Dilworth's sætning, at man kan nøjes med at købe maskiner svarende til det maximale antal ruter med den egenskab, at der ikke er to, som kan betjenes af samme maskine (altså kan lægges i fortsættelse af hinanden).

Om dette så er nok til at redde økonomien i de betrængte flyselskaber, er en anden sag.

10. Opgaver

1. En nødhjælpskonvoj skal levere forsyninger til byerne A, B, C, D og E; vejene mellem disse byer løber i bjergområder, og der er risiko for såvel angreb som miner. Man regner med et tab af lastbiler på de enkelte vejstrækninger som angivet nedenfor:

	A	B	C	D	E
A		2	6	10	5
B	5		4	5	2
C	14	3		9	11
D	11	2	8		2
E	7	5	10	2	

Find den optimale rute og det minimale tabstal.

2. Et ægteskabsbureau har en kundekreds bestående af 7 kvinder og 6 mænd. Fra de udfyldte papirer har man følgende muligheder for kontakter (hvor vi for hver af kvinderne har angivet de mænd, der kan komme på tale som partner):

Kvinder	Mænd
Yvonne	Oluf, Børge, Sigurd
Pernille	Oluf, Hugo
Katrine	Eivind, Børge
Tanja	Børge,Sigurd,Eivind
Camilla	Hugo, Børge
Maria	Jørgen, Eivind, Richard
Louise	Børge, Oluf

Ægteskabsbureauet er overtaget af en fond, der ønsker at finde det gebyr, der netop får udgifter til at balancere med indtægter, samtidig med at betjeningen bliver så god som mulig. Omkostningerne er i indeværende periode på 10.000 kr., og det er fast vedtægt, der betales et fast gebyr på 100 kr. samt et ekstra gebyr ved anvisning af partner. Hvor stort skal dette sidste være?

3. Ved skoleferiens start forventes en betydelig strøm af turister fra København til Euro-Disneyland ved Paris. Jernbamermes kapacitet (i antal tog med plads til 1000 passagerer) er opgjort til:

	Ro	Pu	Pad	Ha	Be	St	Brux	Lie	Pa
Kbhvn	2	11	8						
Ro				3	5				
Pu				14	11				
Pad				10					
Ha					12	12	10	10	
Be						4	4	12	
St							4	12	11
Brux								23	16
Lie									10

(Signaturforklaring: Kbhvn=København, Pu=Puttgarden, Pad=Padborg, Ha=Hamburg. Be=Berlin, St=Strasbourg, Brux=Bruxelles, Lie=Liege, Pa=Paris).

Find det maximale antal rejsende. Giv et forslag til en strækning, hvor en forøgelse af kapaciteten fører til, at der kan komme flere rejsende frem.

4. Løs netværksproblemet givet på tabelform nedenfor:

	A	B	C	D	E	F	G	H	I	J	t
s	4	8	3	2							
A					2	2		12			
B			4				7			5	
C								5			
D					2						
E									1		
F									1	1	
G										10	
H									2	10	
I											2
J											10

11. Litteratur

Algoritmerne til at finde strømme i netværk er det første af en række eksempler (der kommer flere i næste kapitel) på problemer, som egentlig blot er udgaver af lineær programmering og dermed kunne løses med simplex, men hvor der er metoder, som udnytter problemets særlige struktur og derfor er hurtigere. Hovedværket på området er Ford og Fulkerson (1967).

König's sætning (eller rettere det, vi her kalder König's sætning, for der er flere sådanne i litteraturen) kan findes i König (1950).

KAPITEL 10

Lokalisering

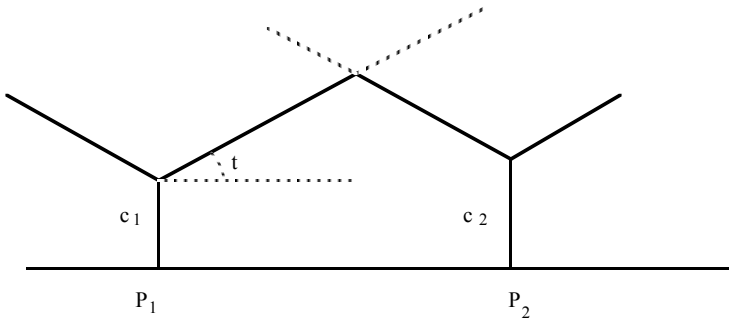
1. Introduktion

Blandt de problemstillinger, som med fordel lader sig formulere og analysere ved hjælp af en præcis model, er der særdeles mange, som har at gøre med produktionens placering og hvad dertil hører i form af transport. At det forholder sig således, er i og for sig ikke særlig underligt, for al økonomisk aktivitet foregår på fysiske lokaliteter; produktionens og forbrugets fordeling på sted og tid er således noget, som egentlig slet ikke kan adskilles fra, hvad man ellers måtte have at sige om sagen. Det har man imidlertid gjort i den økonomiske analyse, og det har været nyttigt i den forstand, at man ved at abstrahere fra tid og sted kunne koncentrere sig om mange andre væsentlige fænomener. Undervejs var det dog som om sted og tid blev skubbet helt ud. Her skal vi rehabilitere stedsaspektet. Det vil ske gennem eksplicit overvejelse dels af *hvor* produktion og forbrug skal placeres, dels ved at inddrag transportomkostningerne når disse placeringer af produktion og forbrug har fundet sted.

Praktikere har til alle tider taget hensyn til lokaliseringsaspektet. Så at sige af sig selv er virksomheder opstået i passende nærhed til råstoffer, energikilder, arbejdskraft eller kunder, alt afhængig af hvad der ved den konkrete produktion vejede tungest. Sådanne common-sense-betragtninger vil næppe slå til i længden, når de lokaliseringsproblemer, der skal løses, bliver passende komplicerede. De kan dog godt være et rimeligt udgangspunkt, og det er det, vi vil bruge dem som i det følgende.

De første systematiske undersøgelser af sammenhængen mellem produktionsstedet for en vare og dens afsætning i geografiske områder stammer fra forrige århundrede. Lad os antage, at en vare produceres på en række forskellige lokaliteter, formaliseret som punkter $P_1, P_2, \ldots, P_k$. I hver af disse produktionssteder er der visse produktionsomkostninger $c_i, i = 1, \ldots, k$; ved transporten ud til forbrugeren løber der endvidere en transportomkostning på; vi antager, at den kan opgøres som et fast beløb t pr. km. (pr. enhed af varen). For den enkelte forbruger af varen gælder det om at købe billigst muligt, *når transportomkostningerne tages med*; det betyder, at forbrugeren i punktet F skal købe hos den producent i, for hvilken

$$c_i + td(F, P_i) \leq c_j + td(F, P_j)$$



Figur 1

for alle $j \in 1, \ldots, k$. De samlede omkostningers afhængighed af afstanden til en given producent er vist i figur 1 (ifølge traditionen kaldes denne figur for "champagneglassene"), og ved hjælp af udtrykket ovenfor kan man nu bestemme hver producents naturlige kundekreds som antydet i figur 1.

I denne problemstilling er det køberens maximeringsproblem, som er det centrale; virksomhederne er allerede placeret i de givne lokaliteter. Men analysen kunne naturligvis udbygges til en overvejelse om, hvorledes den enkelte virksomhed, f.eks. en nyetableret producent, bør placeres for at få den største kundekreds, eller til en generel overvejelse om, hvorledes k virksomheder skal placeres, således at kunderne kan få varen billigst muligt; dette sidste krav er iøvrigt ikke formuleret helt klart; er det den gennemsnitlige kunde eller den værst stillede kunde, vi tænker på? Vi vender tilbage til det senere.

I en lidt simplere formulering – hvor der ses bort fra forskelle i produktionsomkostninger – er bestemmelsen af "naturlig kundekreds" iøvrigt et klassisk matematisk problem, som har fundet nye anvendelser i de senere år.

Lad P være en punktmængde i et vektorrum V. For en given metrik $d(\cdot, \cdot)$ på V kan vi til hvert punkt $z \in P$ bestemme den såkaldte *Voronoi-celle* tilhørende z som

$$V(z) = \{x \in V | d(x, z) \leq d(x, z'), \text{ alle } z' \in V\}.$$

Af speciel interesse er situationen, hvor punktmængden P er stor, således at vi kan tænke os hele rummet fyldt ud med punkter. Der kan endvidere være en vis systematisk i den måde, hvorpå disse punkter er lagt; P kan f.eks. være et *gitter* defineret som alle $\mathbf{Z}$-linearkombinationer af en given punktmængde $(v_1, \ldots, v_k)$.

Klart nok er kundekredsene ovenfor (under forudsætning af ens produktionsomkostninger) et eksempel på Voronoi-celler; disse har dog mange andre anvendelser, hvoraf vi nævner et par (selvom de ikke er specielt operationsanalytiske):

Digital-analog transformation: Ved omsætning af data som har kontinuert variation, til data med diskret variation op står der naturligvis det problem, at der i udgangsmaterialet er (uendelig) mange flere værdier end i det resulterende materiale. Man må derfor erstatte visse værdier med andre, tilnærmede, og som en rimelig

tilnærmelse vil man typisk vælge punktet med mindst afstand (i en passende norm). Dette svarer til at Voronoi-cellen omkring et givet ("digitalt") punkt end alle de data, der tilordnes punktet.

Kommunikationsteori: I den matematiske teori for kommunikation behandler man problemer omkring afsendelse af signaler fra en kilde, som undervejs til modtager udsættes for støj. Hos modtageren opstår der derfor et problem om at rekonstruere signalet, og hvis man kender de potentielt mulige signaler og modtager et eller andet volapyk, vil det være rimeligt at opsøge det nærmeste (igen i passende norm) signal. Dermed får vi en tilordning fra modtagne signaler til opfattede budskaber svarende til Voronoi-cellerne omkring de mulige signaler.

Der er en sammenhæng mellem overvejelser omkring Voronoi-celler og en række andre klassiske problemer; dette kommer specielt klart til udtryk, når man interesserer sig for problemer knyttet til *gitre* i planen. For sådanne punktmængder har det ofte interesse at se på, hvorledes man på pænest måde kan knytte figurer, typisk cirkelskiver, til hvert punkt. Der er mindst tre af den slags tilordninger:

(1) I hvert punkt placeres cirkelskiver med samme radius, således at alle punkter i planen (altså ikke blot gitterpunkterne) er dækket af mindst én cirkel. Dette kaldes en *overdækning,* og problemet er da at finde den mindste radius for sådanne cirkler. I vor givne sammenhæng drejer det sig om at finde en mindste ensartet kundekreds for hver facilitet, således forekomsten af overlappende tilbud (hvor vi forestiller os kunderne spredt ud i hele planen) er så lille som muligt.

(2) Omvendt kan det tænkes, at cirkelskiverne ikke må overlappe. Dette kaldes en *tilpakning* (eng.: packing); det er mindre oplagt at fortolke dette problem i en lokaliseringssammenhæng; det kunne opstå i en situation hvor man var nødt til at tildele alle lokale radiosendere (hvoraf der tænkes at være en i hvert gitterpunkt) samme frekvens. Problemet har en helt naturlig fortolkning i andre forbindelser – overvejelser om hvorledes man pakker kugler i kasser, hvorledes man skærer figurer ud af en træplade osv.

(3) Til sidst har vi så de allerede introducerede Voronoi-celler, der for hvert punkt i planen angiver det nærmeste gitterpunkt; fortolkningen af dem har vi været inde på.

Når det har interesse, at disse tre typer problemer (som iøvrigt vil dukke op i diverse forklædninger adskillige gange) hænger sammen, er det selvfølgelig fordi der så også er rimelig stor chance for, at de metoder, der bruges på et problem, også kan benyttes på de andre. Iøvrigt har sammenhængen også rent teoretisk interesse, som rækker ud over vort emne (nemlig til computer science og matematik).

2. Lokaliseringsproblemer i netværk

I det foregående har lokaliseringsproblemerne udspillet sig i planen $\mathbb{R}^2$, men vi har allerede været inde på, at der var interessante anvendelser af teorien, hvor det kunne være hensigtsmæssigt at vælge en anden dimension end 2, eller måske endda et vektorrum over et endeligt legeme. Naturligvis drejer sagen sig så ikke mere om rent geografiske problemstillinger, men lokalisering er heller ikke det samme som geografi. Det er ikke *afstanden* som sådan, der er interessant, men den *omkostning*, der følger af afstanden. Det er derfor i mange situationer en fordel at abstrahere fra alle de irrelevante aspekter af geografien, således at omkostningsaspektet kommer så meget renere frem.

En rendyrket udgave af lokaliseringsproblemet får man ved at betragte problemet som et *netværk*. Vi har altså et netværk $\mathcal{N}$ bestående af en graf $\Gamma = (V, E)$ og kapaciteter (d_e). Her vil vi foretrække at kalde d_e for *afstanden mellem* v_1 *og* v_2 (hvor $e = (v_1, v_2)$); kanterne er ikke orienterede, og vi har ikke i denne sammenhæng specielt brug for de særlige punkter s og t (fra det forrige kapitel). I vor tolkning som et lokaliseringsproblem vil vi identificere punkter med *kunder*; vi antager at hver kunde k har en given efterspørgsel på w_k enheder (pr. tidsenhed). Hvert punkt er endvidere potentielt mulig som *betjeningssted*.

Lad os se på omkostningerne ved at tilfredsstille kunde k's efterspørgsel, hvis betjeningen sker fra punktet $v \in V$. Hertil skal vi bruge afstanden mellem k og j, $d(k, j)$, der er defineret som det mindste tal

$$\sum_{i=0}^{m} d_{e_i},$$

hvor $(e_i)_{i=0}^m$ er en vej fra j til k, dvs.

$$e_0 = (j, p_1), e_i = (p_i, p_{i+1}), i = 1, \ldots, m-1, e_m = (p_m, k)$$

for en følge $(e_0, e_1, \ldots, e_m)$ af kanter som angivet ovenfor, $+\infty$ hvis der ikke findes nogen vej fra j til k. Vi har da de ønskede samlede omkostninger som $w_k d(j, k)$.

Optimeringsproblemet knyttet til placeringen af betjeningsfaciliteter drejer sig selvfølgelig om at placere faciliteterne således at omkostningerne bliver mindst mulige. Dette er dog ved nærmere eftersyn ikke helt præcist formuleret; hvad forstås ved "omkostningerne"? Vi skal nedenfor give nogle alternative præciseringer af dette:

1. p-median problemet: For et givet tal $p \in \mathbb{N}$ er det opgaven at placere p faciliteter i hvert sit punkt og tilordne kunder til faciliteterne på en sådan måde at de totale omkostninger er minimale.

Vi betegner dette problem med p-MP. Bemærk, at problemet er fuldt karakteriseret ved:

- tallet p
- afstandmatricen hørende til netværket $\mathcal{N}$ med afstandene givet ved $d_e, e \in E$: Denne matrix er $(|V| \times |V|)$-matricen

$$\{d(k,j)\}_{k,j \in V};$$

  – faktisk kan vi her nøjes med at angive tallene d_e,
- for hver kunde $k \in V$, efterspørgslen w_k.

Vi har altså et p-medianproblem p-MP for hver given specifikation af disse data.

Vi kan sige lidt om generelle egenskaber ved løsningen: Der findes altid en løsning til p-MP hvor hver kunde k forsynes fuldt ud fra én facilitet.

Antag nemlig, at faciliteterne er placeret optimalt, og at k er en kunde som forsynes af faciliteterne $j_1, \ldots, j_s$ således at $w_k^{j_i} \geq 0$ er den mængde der ydes fra j_i til k,

$$\sum_{i=1}^{s} w_k^{j_i} = w_k.$$

Vi har da at $d(j_i, k) = d(j_h, k)$, alle i, h, for ellers kunne de samlede omkostninger gøres mindre ved at forsyne k fra den facilitet j, for hvilken

$$d(j,k) = \min_i d(j_i, k),$$

en modstrid mod, at løsningen var optimal. Men da omkostningen ved at betjene kunde k med én enhed er ens for hvert af stederne, kan vi nøjes med at betjene fra ét af dem.

Der er, som man ser, ikke noget særlig dybt i dette argument (der sikrer at løsningen har hvad man kalder "single assignment property"), men det er alligevel af en vis betydning ved konstruktion af algoritmer, da det indskrænker de muligheder, der skal søges igennem for at finde den optimale løsning. I princippet kan denne naturligvis findes ved at *opregne alle muligheder*; man vil dog se, at der hurtigst bliver temmelig mange alternativer, så det er ikke nogen brugbar metode generelt.

2. *p-center problemet*: En anden variant af omkostningsminimeringsproblemet får vi ved i stedet for at betragte *omkostningerne for den værst stillede kunde* (teknisk set *min*imerer vi *max*imum af omkostningerne $w_k d(k,j)$ i stedet for at minimere summen af dem). Vi skal altså her placere de p faciliteter i punkterne $v_1, \ldots, v_k$ således at de samlede omkostninger ved at få w_e leveret (fra punkterne $v_1, \ldots, v_s$), altså

$$\sum_{i=1}^{m} w_k^{j_i} d(k, j_i)$$

er mindst mulige, hvor k er valgt således at udtrykket ovenfor er størst.

Det ses, at også p-center problemet (p-CP) har den såkaldte single assignment property. Beviset går helt som for p-MP; hvis kunde k i optimum forsynes fra mere end en facilitet, da har vi, at k's afstand til samtlige forsynede faciliteter er den samme, og vi kan da forsyne k fra en af dem uden at ændre optimaliteten.

Det betyder dog ikke, at p-MP og p-CP er i alt væsentligt identiske problemer, selvom de er beslægtede. Der er f.eks. en særlig egenskab, den såkaldte *punkt-optimalitet,* som er opfyldt for p-MP og ikke for p-CP. Kort fortalt går det ud på, at man kan overveje at inddrage nye potentielle placeringer ved at oprette nye punkter på de eksisterende kanter. Sådanne udvidede placeringsmuligheder *ændrer ikke* optimaliteten af den oprindelige løsning i p-MP, men de kan godt gøre det i p-CP.

3. ULP (ubegrænset-kapacitets lokaliserings problemet). Her er udgangspunktet et lidt andet end i de foregående problemtyper. Vi har et givet antal *potentielle* faciliteter, men vi kan åbne så mange eller så få af disse, som vi måtte ønske. Der er knyttet visse *faste omkostninger* f_j til åbningen af lokalitet j.

Det er praktisk at formulere dette ved brug af et *todelt* netværk $\mathcal{N}$, dvs. et hvor Γ er todelt, således at mængden V af punkter kan opdeles i to disjunkte mængder V_1 og V_2, hvor hver kant e er en $V_1 - V_2$-kant, dvs. har det ene endepunkt i V_1 og det andet i V_2. Punkterne i V_1 identificeres med kunder, punkterne i V_2 med potentielle faciliteter. Vi har endvidere en afstandsmatrix, der kan skrives som en $|V_1| \times |V_2|$ matrix D med (i,j)-element d_{ij} fortolket som afstanden fra kunde i til (potentiel) facilitet j. Hvis vi med $c_{ij} = w_i d_{ij}$ betegner omkostningerne ved at forsyne kunde i fra facilitet j har vi dermed, at vi skal finde

$$\min_{Q \subset V_2} \left(\sum_{j \in Q} f_j + \sum_{i \in V_1} \min_{j \in Q} c_{ij} \right)$$

over alle ikke-tomme delmængder Q af V_2. Klart nok kan dette problem (som iøvrigt de foregående) i princippet løses ved opregning af alle tilfælde; da der er $2^{|V_2|} - 1$ mulige delmængder Q, ses dette dog hurtigt at blive en kompliceret opgave. Der må derfor bruges andre metoder; vi skal ikke komme nærmere ind på det her.

ULP har et specialtilfælde, som har spillet en prominent rolle, ikke så meget i lokaliseringssammenhæng som i forbindelse med overvejelser om beregningskompleksitet. Lad os se på situationen, hvor tallene i afstandsmatricen D enten er 0 eller ∞ (svarende til at en facilitet j enten ligger helt op ad kunden i, eller der slet ikke er nogen mulig forbindelse). Det viser sig hensigtsmæssigt at omformulere dette ved at indføre $|V_1| \times |V_2|$-matricen A med elementer enten 0 eller 1, således at

$$a_{ij} = \begin{cases} 1 & \text{hvis } d_{ij} = 0 \\ 0 & \text{hvis } d_{ij} = \infty; \end{cases}$$

vi siger at j *dækker* i hvis $a_{ij} = 1$, og kalder delmængden Q af V_2 for en *overdækning* af V_1 hvis

$$\sum_{j \in Q} a_{ij} \geq 1$$

for alle $i \in V_1$.

Problemet *Set Cover* går nu ud på at finde en overdækning af V_1 med færrest mulig elementer; det svarer til, at de faste omkostninger ved at inddrage et j er 1. Dette problem har for det første en række praktiske anvendelser indenfor meget forskellige discipliner, men for det andet spiller det en vigtig rolle i teoretisk sammenhæng. Det kan vises, at Set-Cover er et ret vanskeligt problem (mere præcist, det hører til klassen af NP-komplette problem; herom senere); det var iøvrigt et af de første problemer, for hvilke dette blev fastslået, og det er nyttigt fordi andre problemer ofte lader sig omforme til Set Cover, noget som er væsentligt i forbindelse med overvejelser om beregningskompleksitet.

4. QAP (det kvadratiske tilordningsproblem). I det sidste af vore fire klassiske lokaliseringsproblemer er udgangspunktet en del anderledes end i de foregående. Vi har her *to sæt af data*:

For det første er der givet en mængde $N = \{1, 2, \ldots, n\}$ af *faciliteter* med en afstandsmatrix (af dimension $n \times n$) A, hvor $a_{ij} \geq 0$ angiver afstanden fra facilitet i til facilitet j. Men desuden er der givet en mængde $M = \{a, b, c, \ldots, m\}$ af m forskellige *funktioner* med en *kommunikationsmatrix* B af dimension $m \times m$, hvor $b_{st} \geq 0$ er et udtryk for intensiteten af kommunikationen mellem funktion s og funktion t. Hvis s og t er knyttet tæt sammen, er b_{st} stor; har de intet at gøre med hinanden, er $b_{st} = 0$.

Vort problem er nu at placere funktionerne i faciliteter – én funktion i hver facilitet – således at omkostningerne bliver mindst mulige. Her vil vi ved omkostningerne forbundet med at placere funktion s i facilitet i og funktion j i facilitet t forstå udtrykket

$$a_{ij} b_{st},$$

(således at funktioner med megen kommunikation kommer tæt på hinanden, når omkostningerne minimeres); vi søger altså værdier af variablene

$$x_{is}, i \in N, s \in M,$$

hvor

$$x_{is} = \begin{cases} 1 & \text{hvis } s \text{ placeres i } i \\ 0 & \text{ellers,} \end{cases}$$

og således at vi minimerer

$$\sum_{i \in N} \sum_{j \in N} \sum_{s \in M} \sum_{t \in M} x_{is} x_{jt} a_{ij} b_{st}$$

under bibetingelserne

$$\sum_{s \in M} x_{is} \leq 1 \text{ for alle } i \in N,$$

(højst én funktion i hver facilitet) og

$$\sum_{i \in N} x_{is} = 1 \text{ for alle } s \in M$$

(alle funktioner placeres). Dette er tydelig nok et heltalprogrammeringsproblem. Kriteriefunktionen er kvadratisk i (heltals-)variablene – deraf navnet.

Problemet kan skrives en smule simplere (uden at det dog af den grund bliver nemmere at løse) ved at vi først noterer os, at der ikke er nogen indskrænkning i at antage $m = n$. Hvis $m > n$ har problemet slet ingen løsning, så det kan vi se bort fra. Er $m < n$, kan vi tilføje $n - m$ "tomme" funktioner med kommunikationstal 0 til alle andre funktioner. Givet at vi kan antage $n = m$, behøver vi altså for at finde den bedste placering "blot" at søge over alle permutationer af N, dvs. over alle bijektive afbildninger $\varphi : N \to N$, hvor $\varphi(i)$ er nummeret på den funktion (i en eller anden given nummerering af M), som placeres i facilitet i. Vi kan derfor skrive problemet som

$$\min_{\varphi \in S_n} \sum_{i \in N} \sum_{j \in N} a_{ij} b_{\varphi(i)\varphi(j)}.$$

Bibetingelserne er, som det let ses, indeholdt i kravet om, at vi minimerer over alle φ i mængden S_n af permutationer af $(1, \ldots, n)$.

3. En algoritme til løsning af lokalisering uden kapacitetsbegrænsning

I dette afsnit vil vi se lidt nærmere på en konkret algoritme til brug ved løsning af et af lokaliseringsproblemerne nævnt i afnsit 2, nemlig *ULP* om placering af faciliteter med faste omkostninger til betjening af givne efterspørgsler. Vi skal bruge en udgave af Lagrange-dualitet som beskrevet i kapitlet om heltalsprogrammering.

Problemet er givet som følger: Vi har ialt m faciliteter, som kan åbnes med en omkostning a_i, $i = 1, \ldots, m$. Der skal betjenes en række efterspørgsler w_j hos kunderne $j = 1, \ldots, n$. Transportomkostningerne ved betjening af kunde j fra facilitet i er c_{ij}.

Vi indfører variable $z_i \in \{0, 1\}$, for hver facilitet, hvor z_i er 1 hvis faciliteten er åben, og 0 ellers, og x_{ij} for den transport, der går fra j til i. Problemet kan da

formuleres som følger:

$$\min \sum_{i=1}^{m} a_i z_i + \sum_{i=1}^{m}\sum_{j=1}^{n} c_{ij} x_{ij}$$

under bibetingelserne

$$\sum_{i=1}^{m} x_{ij} = w_j,\ j = 1, \ldots, n,$$

$$w_j z_i - x_{ij} \geq 0,\ i = 1, \ldots, m,\ j = 1, \ldots, n,$$

$$z_i \in \{0, 1\}$$

Her udtrykker bibetingelserne dels det krav, at transporterne til kunder skal tilfredsstille efterspørgslen, dels at der ikke kan transporteres noget ud af en facilitet der ikke er åben (når der sættes $z_i = 0$ i den anden type bibetingelse, får vi $x_{ij} \leq 0$).

Som nævnt vil vi bruge Lagrange dualitet til at håndtere dette problem. Vi tilføjer hver af de første bibetingelser sat til 0, dvs. $w_j - \sum_i x_{ij} = 0$ med en Lagrange multiplikator y_j, så at vi får en ny kriteriefunktion

$$\sum_{i=1}^{m} a_i z_i + \sum_{i=1}^{m}\sum_{j=1}^{n} c_{ij} x_{ij} + \sum_{j=1}^{n} y_j (w_j - x_{ij})$$
$$= \sum_{i=1}^{m} a_i z_i + \sum_{i=1}^{m}\sum_{j=1}^{n} (c_{ij} - y_j) x_{ij} + \sum_{j=1}^{n} w_j y_j;$$

For hvert givet $y = (y_1, \ldots, y_n)$ skal vi minimere denne, idet der nu kun er begrænsningen om at x_{ij} skal være nul hvis z_i er 0. Det vil give os en nedre begrænsning for den optimale kriterieværdi i det oprindelige problem; senere leder vi så efter den bedste af disse begrænsninger.

Holder vi y fast, ser vi, at x_{ij}'erne skal vælges på en bestemt måde for at minimere kriteriefunktionen: Hvis $c_{ij} - y_j$ er positiv, skal x_{ij} være nul, og hvis $c_{ij} - y_j$ er negativ (i dette tilfælde vil vi sige, at transport fra i til j er profitabel) og z_i ikke er 0, skal x_{ij} gøres så stor som mulig, dvs. den skal sættes til w_j (er $z_i = 0$, ved vi, at $x_{ij} = 0$). Det kan vi udtrykke kortere ved at sætte

$$(c_{ij} - y_j) x_{ij} = \min\{0, c_{ij} - y_j\} w_j z_i;$$

det kan nu checkes, at venstre side netop bliver højre side, hvis vi hele tiden vælger x_{ij} ifølge forskriften ovenfor.

Dermed er vi af med x_{ij}'erne i vor Lagrange-funktion. Vi stiler efter at komme af med z_i'erne på tilsvarende måde. Her indfører vi lidt notation, som er nyttig i beregningerne, nemlig størrelsen

$$A_i = -\sum_{j=1}^{n} w_j \min\{0, c_{ij} - y_j\}.$$

Ved hjælp af denne kan vi nu omskrive vor Lagrange-funktion til

$$\sum_{i=1}^{m}(a_i - A_i)z_i + \sum_{j=1}^{n} w_j y_j,$$

og på denne kan vi bruge samme trick som tidligere til at eliminere z_i'erne, givet at vi under alle omstændigheder er ude på at gøre funktionen så lille som mulig: Hvis $a_i - A_i$ er positiv, skal z_i være 0, og er den negativ, skal z_i være 1. Indsættes disse z_i-værdier, får vi ialt udtrykket

$$L(y) = \min\{0, a_i - A_i\} + \sum_{j=1}^{n} w_j y_i$$

for den nedre begrænsning af kriterieværdien, afhængig af den valgte værdi af Lagrange-multiplikatorerne $y_1 \ldots, y_n$. Bemærk, at hvis vi arrangerer os sådan i valget af y_j'er, at der altid gælder $A_i \leq a_i$, da er hele første led 0, og $L(y)$ kan aflæses som $\sum_{j=1}^{n} w_j y_j$. Det kan vi faktisk altid gøre, for hvis der for et eller andet i gælder $A_i > a_i$, så må der være et j med $c_i j < y_j$ (sådan er A_i defineret), og for ethvert sådant j reducerer vi y_j med $\triangle y_j$ ned til c_{ij}; derved falder A_i med $w_j \triangle y_j$, og kriteriefunktionen vokser tilsvarende, men til gengæld skal der også reduceres med $w_j \triangle y_j$, og så går det lige op.

Problemet er nu i første omgang at finde det maximale $L(y)$, når der køres over alle valg af Lagrange-multiplikatorer, så at $A_i \leq a_i$. Det gør vi ved at øge de duale variable, så længe det er muligt. Undervejs benytter vi reglen om at der skal oprettes facilitet, hvis $A_i = a_i$; hvis en kunde får profitable leverancer fra to faciliteter, må vi reducere y_j igen. Der fortsættes indtil der ikke kan øges noget y_j, og den resulterende værdi af $\sum_{j=1}^{n} w_j y_j$ er så den bedste nedre begrænsning. Hvis der til denne svarer en betjening af alle kunder, har vi samtidig en løsning af problemet. Ellers må der bruges andre metoder.

Vi viser metoden først i et meget simpelt tilfælde (eksemplet er hentet fra Erlenkotter, 1976). Der er 5 faciliteter og 8 kunder; kundernes efterspørgsel er 1 (man kan iøvrigt altid få $w_j = 1$ ved at erstatte de oprindelige omkostningsdata for transport af en enhed fra i til j med de samlede omkostninger ved transport af w_j fra i til j). Vi skriver problemet på tabelform som følger:

120	180	100		60		180			100	
210		150	240	55	210	110	155		70	
180	190	110	195	50			195		60	
210	190	150	180	65	120	160	120		110	
170	150	110	150	70	195	200			80	
120	150	100	150	50	120	110	120			

Vi er her startet med den øverste 5×8 tabel, der indeholder omkostningskoefficienterne, og den anden ekstra søjle, som har de faste omkostninger. Der startes nu forneden i en ekstra række, som indeholder de duale variable y_j. De sættes i første omgang til minimum i den pågældende søjle; intuitionen er, at y_j er en pris, som vi kan kræve af kunde j. Det gælder om at forsyne kunderne på en måde, der sikrer størst indtægt målt ved de duale variable, og det kan vi aflæse som rækkesummen. Begrænsningen på vore forsøg på at score kassen er, at vi ikke må kræve en større overpris, end at det kunderne betaler for levering fra enhver given facilitet svarer til transport plus faste omkostninger.

Vi tilføjer nu i hvert felt i tabellen, hvor det er relevant, den overpris i forhold til transportomkostningerne, som vi kræver af kunden (hvis det er en underpris, skrives den ikke). I denne første omgang, hvor vi har valgt minimale priser, kan den aldrig bliver større end 0, og det har vi så skrevet ind de pågældende steder i næste tabel. Første hjælpesøjle udfyldes så med summen af de øverste tal i felterne (stadig 0 i dette tilfælde), og endelig får vi sidste søjle, som angiver z_i, ved at sætte 0 hvis første søjles element er mindre end andens, og 1 hvis de er lig hinanden (første hjælpesøjle må ikke overstige anden). Alt i alt har vi således følgende:

0 120	180	0 100		60		180		0	100	0
210		150	240	55	210	0 110	155	0	70	0
180	190	110	195	0 50			195	0	60	0
210	190	150	180	65	0 120	160	0 120	0	110	0
170	0 150	110	0 150	70	195	200		0	80	0
120	150	100	150	50	120	110	120			

Dette er start-tabellen eller, om man vil, det 0'te skridt i algoritmen. Vi begynder nu at opdatere søjlerne, hvilket sker ved at gå til den næstlaveste af de omkostninger, der er angivet i hver søjle, idet vi starter med søjle 1. Her ændrer vi y_1 fra 120 til 170, og de respektive overpriser skrives ind i øverste hjørne. Så går vi videre til næste søjle; undervejs bør vi holde øje med, at summen af vore overpriser fra given kilde (dvs. tallene i A_i-rækken) ikke overstiger de faste omkostninger.

I vort eksempel kan vi gå hele tabellen igennem, således at den nye opdaterede tabel kommer til at se således ud:

50 120	0 180	10 100		60		180		60	100	0
210		150	240	0 55	210	50 110	0 155	50	70	0
180	190	0 110	195	5 50			195	5	60	0
210	190	150	0 180	65	75 120	0 160	35 120	110	110	1
0 170	30 150	0 110	30 150	70	0 195	200		60	80	0
170	180	110	180	55	195	160	155			
			*		*	*	*			

Der er nu kommet en facilitet i række 4, hvor A_i er blevet lig a_i. De stjernede søjler er *blokerede:* prisen ikke kan sættes i vejret uden at der kommer knas med A_i i fjerde række.

Vi fortsætter nu med nok en runde. Der opdateres fra venstre, idet vi undervejs holder øje med, om det nu også kan lade sig gøre. Hvis en søjle ikke kan opdateres uden at der kommer ubalance, springer vi den over og går videre. På denne måde får vi opdateret søjlerne 1, 2 og 5, og den nye tabel ser derefter således ud:

60 120	10 180	10 100		0 60		180		80	100	0
210		150	240	5 55	210	50 110	0 155	55	70	0
0 180	0 190	0 110	195	10 50			195	10	60	0
210	0 190	150	0 180	65	75 120	0 160	35 120	110	110	1
10 170	40 150	0 110	30 150	70	0 195	200		80	80	1
180	190	110	180	60	195	160	155			
*	*	*	*		*	*	*			

Der er nu også kommet en facilitet i række 5, og vi kan gå videre til næste runde. Det er kun søjle 5, hvis pris kan øges, og gør vi det får vi den endelige løsning:

Der er nu fundet en løsning: Kunderne skal betjenes fra de to sidste faciliteter, og hvis der er valg (som for kunderne 4 og 6), skal det ske gennem den billigste transportvej. Der bliver dermed overensstemmelse mellem summen af priserne i nederste række og omkostningerne til såvel transport som oprettelse af faciliteter (check det!).

60 120	10 180	10 100		5 60		180		85	100	0
210		150	240	10 55	210	50 110	0 155	60	70	0
0 180	0 190	0 110	195	15 50			195	15	60	0
210	0 190	150	0 180	0 65	75 120	0 160	35 120	110	110	1
10 170	40 150	0 110	30 150	70	0 195	200		80	80	1
180	190	110	180	65	195	160	155			

* * * * * * * *

Eksemplet er forholdsvis pænt i den forstand, at proceduren går uproblematisk til et optimum. Vejen mod den bedste undergrænse kan generelt være mere kompliceret, og den bedste undergrænse kan være lavere end minimum i det oprindelige problem.

4. Litteratur

Dette kapitel er, som man kan se fra overskriften, knyttet sammen ved at den omhandler modeller til løsning af lokaliseringsproblemer af forskellig art. Metoderne hertil er overvejende af den type, som blev behandlet i kapitlet om diskret optimering, hvortil der henvises for yderligere læsning.

KAPITEL 11

Transport og assignment

1. Introduktion

Tæt knyttet til overvejelserne omkring lokalisering er problemer om at flytte varer med billigste omkostninger. Faktisk er netop transportomkostningerne den centrale faktor i spørgsmål om lokalisering. Men der kan også være transportproblemer, hvor det *ikke* er spørgsmålet om placering af faciliteter, der er væsentligt, idet disse er givne, men hvor det drejer sig om at finde den rigtige måde at sende varer mellem afsendere og modtagere.

Disse *transportproblemer* er givne ved, at der foreligger en række

$$\textit{kilder } s_1, \ldots, s_m$$

og tilsvarende en række

$$\textit{terminaler } t_1, \ldots, t_n.$$

Der kan transporteres varer fra enhver kilde s_i til enhver terminal t_j; omkostningerne herved er c_{ij} pr. transporteret enhed. Endelig er der givet en *beholdninger* $p_i \geq 0$ i kilden s_i $i = 1, \ldots, m$, og der er givet efterspørgsel $q_j \geq 0$ i hver terminal t_j. Det antages at

$$\sum_{i=1}^{m} p_i = \sum_{j=}^{n} q_j;$$

vi er altså her kun interesseret i transporten, mens eventuelle uoverensstemmelser mellem efterspørgsel og beholdning holdes udenfor.

Bemærk, at vi egentlig har at gøre med strøm i et netværk. Netværket er i forhold til det forrige kapitel en smule mere kompliceret i den forstand, at der er flere kilder og terminaler; til gengæld er det så meget mere simplere med hensyn til den underliggende graf, idet vi har, at enhver kilde er forbundet med en kant til enhver terminal, og der er ingen punkter udover disse. Det kan således dårligt betale sig at køre vort apparatur fra forrige kapitel i stilling (selvom det egentlig er sådanne metoder, vi skal benytte).

Det klassiske transportproblem går ud på at finde en transportplan

$$x_{ij},\ i = 1, \ldots, m,\ j = 1, \ldots, n,$$

således at alle efterspørgsler tilfredsstilles, og således at omkostningerne bliver mindst mulige. Dette kan klart nok formuleres som et LP-problem:

$$\min \sum_{i,j} c_{ij} x_{ij}$$

under bibetingelserne

$$\sum_{i} x_{ij} = q_j,\ i = 1, \ldots, m,$$

$$\sum_{j} x_{ij} = p_i,\ j = 1, \ldots, n,$$

$$x_{ij} \geq 0,\ \text{alle } i, j.$$

Da vi har at gøre med et vanligt lineært programmeringsproblem, er der sådan set ikke noget i vejen for at løse ved simplex-metoden. Det vil dog ofte være ret store tableau'er, man får med at gøre, idet der jo er mn variable og $m + n$ bibetingelser. Heldigvis findes der særlige metoder til disse problemer, der udnytter den struktur, problemet har. Det skal vi se lidt nærmere på i det følgende.

Transportalgoritmen. Vi gennemgår en metode til at finde en løsning til transportproblemet; algoritmen består af tre trin, nemlig

(1) finde en brugbar løsning,

(2) teste løsningen for optimalitet,

(3) forbedre løsningen, hvis den er inoptimal.

De tre trin vil blive diskuteret hver for sig i det følgende. Fælles for de tre trin er, at der arbejdes med problemet stillet på skemaform som en $(m \times n)$-matrix af omkostningskoefficienterne c_{ij} suppleret med henholdsvis en søjle og en række (til venstre og neden for matricen), hvor beholdninger og efterspørgsler er placeret:

p_1	c_{11}	$\cdots$	c_{1n}
$\vdots$	$\vdots$		$\vdots$
p_m	c_{m1}	$\cdots$	c_{mn}
	q_1	$\cdots$	q_n

Undervejs i algoritmen vil skemaet også blive brugt til andre beregninger.

Trin 1. Ved en brugbar løsning forstås her som normalt et sæt af værdier af variablene x_{ij}, der opfylder bibetingelserne. For at finde en sådan løsning indfører vi en *ordning* af de n søjle svarende til terminalerne. Denne ordning sker efter størrelsen

$$q_j(\max_i c_{ij} - \min_i c_{ij}),$$

der er et udtryk for den maximale besparelse opnået ved at flytte den samlede leverance til j fra dyr til billig transportvej.

Hvis j^0 er søjlen med størst værdi af denne besparelse (som indtil videre er rent beregningsteknisk), udvælger vi denne søjle og vælger x_{ij^0} således, at q_{j^0} tilfredsstilles på billigst måde. Der fortsættes med søjlen q_{j^1} osv. *indtil en kilde er udtømt*. Herefter fjernes de søjler, hvis efterspørgsel er tilfredsstillet, og den kilde, der er udtømt, og der fortsættes på samme måde med det reducerede skema (hvor kildernes beholdning naturligvis er formindsket med de transporter, der er tildelt værdier). Proceduren vil ende med at have behandlet samtlige søjler, hvorefter der foreligger en brugbar løsning.

Eksempel 11.1

Antag at vort transportproblem har følgende data:

4	2	11	10	3	7
8	1	4	7	2	1
9	3	9	4	8	12
	3	3	4	5	6

For at finde den søjle, der skal startes med, udregner vi besparelserne. De er:

6	21	24	30	66

Vi kan nu starte med at tildele transporter til terminal 5; det skal klart nok være fra kilde 2, idet anden række har den mindste omkostningskoefficient. Samtlige 6 enheder til denne terminal kommer da fra kilde 2, så vi har $x_{25} = 6$ og $x_{i5} = 0$ ellers.

Ingen kilde er endnu udtømt, så vi fortsætter med søjle 4, som har næststørste besparelse. Igen er det billigst at forsyne fra kilde 2, men denne er udtømt efter 2 enheder. De resterende enheder må da komme fra kilde 1, hvorfra transporten til terminal 4 er næstbilligst. Der bliver altså sendt 3 enheder fra kilde 1 til terminal 4.

Vi har nu klaret terminal 4 og 5, hvorved vi har udtømt kilde 2 og reduceret kilde 1 med 3 enheder. Det er ensbetydende med, at vi står overfor et reduceret transportproblem – med to kilder og tre terminaler – af følgende form:

1	2	11	10
9	3	9	4
	3	3	4

For dette problem finder vi de tilhørende besparelser:

3	6	24

Tydeligt nok er det den sidste søjle, som rangerer øverst mht. besparelse; vi sender derfor 4 enheder fra kilde 3 (den billigste) til terminal 3, hvorefter der fortsættes med søjle 2, der også skal forsyneds fra kilde 3. Der resterer nu 3 enheder til terminal 1, svarende til de 3 enheder, der er tilbage.

Eksempel 11.1, fortsat

Ialt får vi følgende brugbare løsning til transportproblemet:

1	0	0	3	0
0	0	0	2	6
2	3	4	0	0

De samlede transportomkostninger ved denne løsning er $T_C = 70$.

Det bemærkes, at denne løsning har temmelig mange 0'er. Det er ret væsentligt for den videre procedure, at der ikke er for mange af dem; vi skal generelt bruge $m + n - 1$ elementer forskellige fra nul for at kunne komme videre (det har vi heldigvis her).

Vi har nu to problemer tilbage, nemlig dels at checke løsningen for optimalitet, dels at finde en bedre løsning. For at checke optimalitet går vi frem på følgende måde:

Trin 2. Der opstilles en tabel svarende til den oprindelige $m \times n$ matrix, med omkostningerne c_{ij} placeret på de pladser, der svarer til elementer i løsningen, som ikke er 0. Der tilføjes tom søjle z til sidst og en tom række y i bunden.

Et vilkårligt element i enten y eller z sættes til nul; derefter bestemmes alle de endnu ukendte elementer i den oprindelige tabel samt i y og z ved betingelsen

$$c_{ij} = y_i + z_j, \text{ alle } i, j.$$

Denne regel bestemmer entydigt de resterende elementer i matricen, ihvertfald hvis den brugbare løsning, som checkes, har mindst $m + n - 1$ elementer forskellig fra 0. Imodsat fald kaldes løsningen *degenereret,* og der gås frem på særlig måde, som vi kommer tilbage til senere. Den færdioge $(m \times n)$-matrix (den ekstra søjle og række er ikke interessant mere) kaldes H.

Vi tager nu matricen H og trækker omkostningsmatricen C fra. Derved fås en matrix L, hvor der ifølge vor konstruktion står 0 på alle pladser, der svarer til ikke-nul elementer i løsningen. De resterende elementer kan være negative, nul eller positive. *Hvis der findes positive elementer blandt dem, er den betragtede løsning ikke optimal*.

Eksempel 11.2

I vort eksempel fra før skal vi starte med at opstille følgende tabel:

2			**3**		
			2	**1**	
3	**9**	**4**			
0					

Eksempel 11.2, fortsat

Vi har placeret et 0 i nederste række; det kunne være placeret hvorsomhelst i den tomme række eller søjle. Vi fortsætter nu med at fylde tabellen ud under den betingelse, at elementet i række i og søjle j skal være summen af, hvad der står i nederste række, yderste søjle. Herved får vi:

2	8	3	**3**	2	2
1	7	2	**2**	**1**	1
3	**9**	**4**	4	3	3
0	6	1	1	0	

Efter at have gjort dette, sammenligner vi matricen af de første m rækker og n søjler med omkostningsmatricen. Det ses, at der er et positivt element i differensmatricens 2. række, 2. søjle, nemlig 3. Vi konkluderer, at den betragtede løsning ikke er optimal.

Trin 3. Hvis optimalitetschecket viste, at en løsning ikke var optimal, blev der samtidig fundet pladser i differensmatricen L, hvor de tilhørende elementer l_{ij} var positive; vælge i^0 og j^0, så at elementet $l_{i^0j^0}$ er det største af disse positive elementer.

Vi laver nu en *rundtur* fra position (i^0, j^0) blandt de positioner (i, j), som svarer til ikke-nul elementer i den givne løsning x. Det sker ved, at vi fra punktet (i^0, j^0), som vi betegner P_0, bevæger os til nærmeste x-element i samme række i^0, således at den tilhørende søjle har to x-elementer, dette punkt kalder vi P_1, fra P_1 i samme søjle til det andet x-element P_2, derefter igen videre i P_2's række på samme måde som tidligere (til nærmeste endnu ubrugte x-element, hvis søjle har endnu et x-element), og så fremdeles. De enkelte x-elementer må naturligvis kun bruges én gang; til gengæld regnes P_0 med blandt dem, og derved vil rundturen før eller senere ende med P_0.

Når vi har en sådan rundtur, giver vi punkterne fortegn henholdsvis $+$ og $-$ startende med P_0 (det kommer automatisk til at passe med, at sidste punkt før P_0 har $-$). Vi finder dernæst mindste tal ξ blandt tallene x_{ij} hørende til de punkter, der har fortegn $-$, og den oprindelige løsning modificeres nu ved at tildeles ξ på alle pladser, hvor der står $+$, og trækkes ξ fra overalt hvor der står $-$.

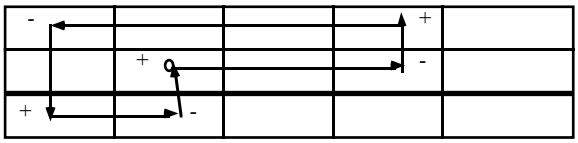
På grund af den måde, vi har konstrueret rundturen på, er den modificeres løsning stadig brugbar. Den kan nu checkes for optimalitet, og der kan fortsættes.

Degenererede løsninger. Det eneste, der mangler i vor transportalgoritme, er en forskrift for, hvad vi gør i det tilfælde, hvor der ikke er $m+n-1$ elementer forskellig fra 0 i en brugbar løsning. Der kan efter omstændighederne være adskilligt færre ikke-nul elementer.

Hvis dette skulle forekommes, ændres fremgangsmåden i algoritmens trin 2: Først forsøges så mange som muligt af tabellens positioner udfyldt, idet der som sædvanlig startes med et 0 et sted i den nederste række eller den sidste søjle. Når dette stopper, vil der stadig være tomme felter i såvel tabellens indre som yderste række eller søjle. I et af disse tomme felter skriver vi nu den oprindelige

Eksempel 11.4

I vort gennemgående eksempel er positionen $(2, 2)$ faktisk den, der giver det maximale element i differensmatricen, og vi laver derfor en rundtur herfra. Den kommer til at se ud som på figur 1, hvor vi også har angivet fortegn såvel som det tal, der af den hiditige løsning blev foreskrevet til den pågældende plads.



Efter modifikation i overensstemmelse med reglerne ovenfor fås nu en ny løsning

0	0	0	4	0
0	1	0	1	6
3	2	4	0	0

De samlede transportomkostninger ved denne løsning er $T_C = 67$.

Denne løsning er iøvrigt optimal; det kan ses ved at gennemføre proceduren fra trin 2, hvor man får en differensmatrix med alle elementer (pånær dem, der stammer fra løsningen) negative. Det sidste (negative snarere end blot ikke-positive elementer) betyder iøvrigt, at løsningen er *entydig*.

omkostningskoefficient, hvorefter vi kan gå videre med den sædvanlige procedure. Dette svarer til, at den brugbare løsning ændres ved, at der tilføjes en meget lille transport ε_1 fra en af kilderne til en af terminalerne; det indføjer man i sin løsning. Skulle forsøget på at fylde tabellen ud gå i stå nok en gang, gør man det samme, der sættes koefficienter ind på en plads, hvor der tilsvarende i løsningen placeres endnu et ε_2, og så fremdeles.

Derved ødelægger man strengt taget brugbarheden af løsningen; men hvis den nye "næsten-brugbare" og ikke-degenererede løsning alligevel viser sig ikke at være optimal (mht. omkostninger), vil man alligevel skulle modificere den i næste trin.

Der fortsættes nemlig nu med den nye løsning (med ε'erne) på samme måde som før, dvs. med optimalitetschecket i trin 2 og forbedringen i trin 3. Hvis der ikke er positive elementer i differensmatricen, er den oprindelige løsning (uden ε'er) optimal. Hvis der er positive elementer, bruges den største til at lave en rundtur, og løsningen modificeres i overensstemmelse med denne. Det kan godt ske, at det undervejs kun er et af de kunstige ε'er, der skiftes rundt i løsningen. Det skal man ikke tage sig af, det ender, som det skal alligevel.

2. Teorien bag transportalgoritmen

Som vi har set, består algoritmen til løsning af transportproblemer af tre dele, nemlig

(i) Finde en brugbar løsning (med højest $m + n - 1$ ikke-nul elementer),

(ii) Optimalitetscheck,

(iii) Opdatering af løsning.

Hver af disse beregningstrin kan umiddelbart forekomme mystiske, men de er det egentlig ikke. Der er blot tale om en algoritme til løsning af et specielt LP-problem, hvor man kan komme let igennem ved at bruge en slags slingrekurs mellem det oprindelige og det duale problem.

At der er mening med tingene, vil fremgå at det følgende. Vi tager beregningstrinene et ad gangen:

Trin (i): Vi skal finde en brugbar løsning, og det er i princippet ikke nogen særlig svær opgave, givet at summen af efterspørgslerne er lig summen af beholdningerne i kilderne. Det betyder nemlig, at der ihvertfald findes en brugbar løsning (man kan jo bare starte fra øverste venstre hjørne og dele ud, så længe der er noget at give af og terminalen ikke har fået nok). I samme åndedrag noterer vi os lige – det er nemlig godt at vide senere hen - at der faktisk findes en optimal plan (mængden af brugbare planer er kompakt og den er ikke-tom, har vi lige set; så antager omkostningsfunktionen sit minimum et eller andet sted). Vi risikerer altså ikke at skulle lede efter noget, der ikke findes.

Der er den hage ved en alt for intuitiv tilgang til at finde brugbare løsninger, at vi vil have en, som er nul for alle pånær højest $m+n-1$ af de ialt mn variable x_{ij}. Hvorfor vi insisterer på det, ser vi under næste punkt. Indtil videre er det bare et krav stillet til vor fremgangsmåde.

Selv denne betingelse lader sig ret let opfylde, men det er rart at have en sikker forskrift, og den, vi bruger, er sikker. Det viser vi lige:

Vi bruger et induktionsbevis efter summen af rækker og søjler i tabellen af omkostningskoefficienter. Denne sum er ≥ 2; hvis den er 2, har vi det ret simple transportproblem med kun en kilde og en terminal; der er kun en løsning, som både er brugbar og optimal, og der er selvfølgelig højest ét ikke-nul element i løsningen (for der er kun en variabel). Lad os så antage, at vi har vist, at vor procedure giver højest $m+n-1$ ikke-nul elementer i løsningen for alle problemer med $m+n<k$ for et vist $k>2$, og at vi nu har et problem med $m+n=k$.

Vi kører efter bogen, finder spændene og begynder at tilfredsstille den bedst rangerede terminal. Når vi nu første gang tildeler så meget som muligt fra den billigste kilde sker der en af to ting: Enten udtømmer vi denne kilde, og problemet reducerer til et nyt med en række mindre, eller vi tilfredsstiller terminalen fuldt ud, hvorved problemet reduceres til et med en søjle mindre. I begge tilfælde arbejder vi videre med et transportproblem, som har summen af rækker og søjler en mindre end det oprindelige problem. Her gælder induktionsantagelsen, så der er en løsning til dette med højest $m+n-2$ ikke-nul elementer. Hertil lægger vi den tildeling, vi startede med, og har ialt en løsning til det oprindelige problem med ikke over $m+n-1$ ikke-nul elementer.

Trin (ii): Optimalitetschecket bygger på, at det oprindelige problem har et dualt, som på visse måder er nemmere at håndtere.

For at gennemskue dette skriver vi lige transportproblemets bibetingelser på matrixform. De kommer til at se således ud

$$\begin{pmatrix} 1 & 1 & \dots & 1 & 0 & 0 & \dots & 0 & 0 & \dots & \dots & 0 \\ 0 & 0 & \dots & 0 & 1 & 1 & \dots & 1 & 0 & \dots & \dots & 0 \\ & & \dots & & & & \dots & & & \dots & \dots & \\ & & \dots & & & & \dots & & & \dots & \dots & \\ 1 & 0 & \dots & 0 & 1 & 0 & \dots & 0 & 1 & \dots & \dots & 0 \\ 0 & 1 & \dots & 0 & 0 & 1 & \dots & 0 & 0 & \dots & \dots & 0 \\ & & \dots & & & & \dots & & & \dots & \dots & \\ & & \dots & & & & \dots & & & \dots & \dots & \end{pmatrix} \begin{pmatrix} x_{11} \\ x_{12} \\ \vdots \\ x_{1n} \\ x_{21} \\ \vdots \\ \vdots \end{pmatrix} = \begin{pmatrix} s_1 \\ \vdots \\ s_m \\ d_1 \\ \vdots \\ d_n \end{pmatrix}.$$

Systematikken er her, at de første m bibetingelser, der går ud på at ikke må sendes mere ud af en kilde i, end der er, drejer sig om at summere alle x_{ij} med i på første plads; derfor får vi en række med 1-taller på n hinanden følgende pladser. De sidste n bibetingelser går på terminalerne; her skal der summes over første koordinat i x_{ij}, så vi får vektorer, hvor der står 1 hver n'te gang og ellers 0.

Når vi laver det duale, skal vi for det første maximere i stedet for at minimere. Lad os kalde de duale variable for $u_1, \dots, u_m$ og $v_1, \dots, v_n$ svarende til søjlen på højre side overfor. Kriteriefunktionen er altså

$$\sum_{i=1}^{m} s_i u_i + \sum_{j=1}^{n} d_j v_j,$$

som vi maximerer under bibetingelserne

$$\begin{pmatrix} u_1 \\ \vdots \\ u_m \\ v_1 \\ \vdots \\ v_n \end{pmatrix}^t \begin{pmatrix} 1 & 1 & \dots & 1 & 0 & 0 & \dots & 0 & 0 & \dots & \dots & 0 \\ 0 & 0 & \dots & 0 & 1 & 1 & \dots & 1 & 0 & \dots & \dots & 0 \\ & & \dots & & & & \dots & & & \dots & \dots & \\ & & \dots & & & & \dots & & & \dots & \dots & \\ 1 & 0 & \dots & 0 & 1 & 0 & \dots & 0 & 1 & \dots & \dots & 0 \\ 0 & 1 & \dots & 0 & 0 & 1 & \dots & 0 & 0 & \dots & \dots & 0 \\ & & \dots & & & & \dots & & & \dots & \dots & \\ & & \dots & & & & \dots & & & \dots & \dots & \end{pmatrix} \leq \begin{pmatrix} c_{11} \\ c_{12} \\ \vdots \\ c_{1n} \\ c_{21} \\ \vdots \\ \vdots \end{pmatrix}^t$$

(tegnet ^{t} står for, at søjlerne er transponeret; det er egentlig rækker). Det ser formidabelt ud, men det snyder lidt; ganger vi det stygge matrixprodukt ud, får vi bare ulighederne

$$u_i + v_j \leq c_{ij}$$

for alle i, j (prøv!). Da det oprindelige problem var givet ved lighedstegn, er der ikke nogen betingelser om ikke-negativitet på de duale variable.

Nu vender vi tilbage til vort optimalitetscheck. Lad os nemlig antage, at vi har en brugbar løsning (x_{ij}) til transportproblemet, og at vi har fået opgivet nogle

værdier u_i, v_j af de duale, som opfylder bibetingelserne ovenfor. Så er de altså brugbare løsninger. Antag endvidere, at de opgivne værdier er sådan at der gælder lighedstegn

$$u_i + v_j = c_{ij}$$

for de (i, j) for hvilke $x_{ij} \neq 0$. Så kan vi opgøre de samlede omkostninger som

$$\begin{aligned}\sum_{i,j} c_{ij}x_{ij} &= \sum_{i,j}(u_i + v_j)x_{ij} \\ &= \sum_i \sum_j (u_i + v_j)x_{ij} = \sum_i u_i(\sum_j x_{ij}) + \sum_i \sum_j v_j x_{ij} \\ &= \sum_i s_i u_i + \sum_j \sum_i v_j x_{ij} = \sum_i s_i u_i + \sum_j v_j d_j.\end{aligned}$$

Der er manipuleret lidt i denne udledning: først erstattede vi c_{ij} med $u_i + v_j$; det går godt, for hvis der ikke gælder lighedstegn, er det tilhørende x_{ij} alligevel lig nul. Så nusser vi lidt med dobbeltsummen; det kommer ud på at summe i en smart rækkefølge, så at man kan bruge det man ved og summerne af x_{ij} over et af fodtegnene.

Når vi først har dette udtryk, kan vi se, at de duale variable er både brugbare og optimale, for et vilkårligt sæt duale variable vil altid give en kriterieværdi, som er $\leq$ den mindst mulige i det oprindelige program; er de lig hinanden, må vi være i optimum (dette er intet andet end hovedsætningen om lineær programmering, så vi bliver ikke synderlig overraskede).

Dermed har vi begrundelsen for det umiddelbart noget mystiske optimalitetscheck. Det drejer sig blot om at konstruere sig et sæt u_i'er og v_j'er, der opfylder betingelserne om lighedstegn der hvor x_{ij} ikke er nul, og som er $\leq$ ellers. Lykkes dette, er man i optimum.

Hvad så med det indledende 0? Og de lidt underlige ϵ'er, som man kan få brug for? Det er egentlig af mindre betydning her, men det bliver væsentligt i næste trin, så lad os tage denne sag med:

Når man skærer igennem de maleriske detaljer, er transportalgoritmen egentlig blot en slags stærkt tilpasset simplex-metode. Man kører faktisk fra basisløsning til basis-løsning, og man skifter en variabel ind i basis hvis den giver et godt bidrag til optimum. Det ser vi om lidt. Men referencen tilbage til simplex og til basisløsninger giver straks forklaringen på, at vi har brug for en løsning med kun $m + n - 1$ ikke-nul elementer; sådan ser basisløsningerne nemlig ud. Der er ganske vist $m + n$ rækker i bibetingelsesmatricen, men de er ikke uafhængige! Man kan f.eks. udlede den sidste fra de øvrige: En transportplan, som opfylder alle bibetingelser pånær muligvis den om at leverancerne til sidste terminal skal være lig efterspørgslen, må nødvendigvis også opfylde denne sidste betingelse (check selv). Nu ved vi, hvorfor det var vigtigt at finde den første transportplan på den lidt specielle måde; den sikrede os en basisløsning.

Hvis der er færre end $m + n - 1$ ikke-nul elementer, må vi supplere op med variable, som så kommer med i basis på nul-niveau. Det kan man også komme ud for i simplex, hvor det dog ikke optræder så ofte som lige her. I transportalgoritmen er der tradition for, at man markere dette nul-niveau med et ε, der altså er så lille, at det faktisk er nul. Når man alligevel har det med, er det fordi en basis skal have $m+n-1$ variable; det er da bedre at skrive ε end 0, så man ikke selv bliver forvirret.

Trin (iii): Med al vor viden fra det foregående bør vi ikke blive overraskede over, at vi interesserer os for det element i tabellen, hvor forskellen mellem vor udregnede $u_i + v_j$ og c_{ij} er størst mulig (og positiv hvis optimalitetschecket slog fejl). Det svarer faktisk til at vi i simplex opsøger den mest interessante ikke-basisvariabel ved at inspicere koefficienterne i c-rækken.

Vi skal nu have denne variabel ind i vor basis. I simplex-algoritmen laver man et pivotstep; vor rundtur er faktisk en tilsvarende historie: Hvis vi skal give den nye variabel x_{ij} positiv vægt, må vi jo reducere strømmen fra kilde i tilsvarende et andet sted. Dette andet sted skal være sådan, at den terminal, som nu får mindre fra i, kan få mere fra en anden kilde; da den nye kilde skal yde mere til terminalen, må der reduceres til en anden osv., indtil vi er tilbage ved udgangspunktet. Det er fra denne beskrivelse oplagt, at en sådan omdirigering altid er mulig; man risikerer altså ikke, at der ikke kan findes nogen rundtur.

Når vi derefter finder den mindste strøm, der kan omdirigeres på denne måde, svarer det til sammenligning af b-række og ny basisvariabel-række i simplex-tableauet. Nok engang er der er intet nyt under solen, det er blot kommet til at se lidt anderledes ud, og det er blevet lettere at håndtere. Teknisk sparer man en lang række irrelevante beregningstrin (affødt af de mange nuller i bibetingelsesmatricen). En sådan besparelse er værd at tage med; den er særdeles følelig, når problemerne er tilstrækkelig store.

3. Assignment-problemet

Tæt knyttet til transportproblemet – ihvertfald hvad angår den formelle struktur – er det såkaldte *assignment-problem*. Vi har her et vist antal, n, arbejdere, der hver er i stand til at udføre det samme antal, dvs. n forskellige jobs. Vi antager, at vi har et mål for arbejder i's præstation på job j i form at en karakter eller et pointtal r_{ij}. Opgaven er nu at anvise præcis ét job til hver arbejder, og at gøre det på en sådan måde, at det samlede pointtal bliver størst muligt.

Umiddelbart drejer dette sig om at vælge en bestemt bland alle de $n!$ måder, på hvilke man kan anvise n jobs til n arbejdere. Men hvis n er et stort tal, bliver tallet $n!$ ganske enormt, og det er praktisk uigennemførligt at søge alle disse kombinationer igennem.

I stedet kan man da formulere opgaven som et maximeringsproblem: Vi søger

en matrix

$$X = \begin{pmatrix} x_{11} & \dots & x_{1n} \\ \vdots & & \vdots \\ x_{n1} & \dots & x_{nn} \end{pmatrix}$$

hvor hvert enkelt element x_{ij} er enten 0 eller 1, således at

$$T_R = \sum_{i,j} x_{ij} r_{ij}$$

er maximal, når *der samtidig skal gælde,* at i hver række og i hver søjle i X er der kun ét ikke-nul element.

Umiddelbart virker det ikke, som om der herved er opnået nogen lettelse, for vi skal stadig lede efter løsninger med særlige egenskaber – her en matrix, hvis ikke-nul elementer er placeret på en ganske bestemt måde. Det er dog alligevel en god indgangsvinkel; pointen er, at vi har fået vort problem omformet til noget, vi har metoder til at løse.

Mere om König's sætning. I kapitel 9 diskuterede vi todelte grafer og fik blandt meget andet et resultat om todelte grafer, den såkaldte König's sætning (sætning 3). Dette – tilsyneladende ret abstrakte – resultat viser sig at være nøglen til en algoritme til løsning af assignment problemet, nemlig Kuhn's *ungarske metode* (opkaldt således af Kuhn, fordi König var ungarer.

For at få dette frem skal vi en lille omvej; ved *strukturrangen* af en matrix A forstås det største tal r, således at der findes elementer forskellige fra 0, der står i r forskellige rækker og r forskellige søjler i matricen A. Matricen

$$\begin{pmatrix} 3 & 0 & 0 & 0 \\ 0 & 0 & 2 & -1 \\ 0 & 0 & 1 & -0.5 \\ 2 & 0 & 0 & 0 \end{pmatrix}$$

har strukturrangen 3, for der er ikke-nul elementer på pladserne $(1,1)$, $(2,3)$ og $(3,4)$ (hvor det ses, at ingen række eller søjle optræder mere end én gang), men vi kan ikke finde 4 tilsvarende positioner. Det bemærkes i farten, at den sædvanlige rang af matricen ikke svarer til strukturrangen; i vort eksempel er der kun 2 lineært uafhængige søjlevektorer, så rangen er 2.

Vi har brug for en omformulering af König's sætning:

Strukturrangen af en matrix A er lig med det mindste antal rækker og søjler i A, der tilsammen indeholder alle fra nul forskellige elementer i A.

For at bevise dette betragtes den todelte graf Γ, hvis punktmængde er $V = V_1 \cup V_2$, hvor $V_1 = 1, \dots, m$ er de m rækker i A og $V_2 = 1, \dots, n$ de n søjler i A. og hvor to punkter er forbundet med en kant netop når $a_{ij} \neq 0$. Vi har da, at strukturrangen netop er $|\mathcal{M}|$, hvor $\mathcal{M}$ er en maximal parring i Γ.

Vi har nu, at dette tal $|\mathcal{M}|$ er lig med størrelsen af den mindste overdækning i grafen, hvilket vil sige den mindste mængde af punkter, således at enhver kant er incident med et punkt fra mængden. Men da punkter netop er rækker eller søjler, svarer dette til at vælge det mindste antal rækker og søjler, så at alle ikke-nul elementer er i en af disse rækker og søjler. □

I vort eksempel har vi, at vi f.eks. kan dække alle ikke-nul elementer ved at bruge første, tredie og fjerde søjle i matricen; vi kan altså klare os ved søjler alene, men man kan godt komme ud for at skulle bruge begge dele.

Således som sætningen ovenfor er formuleret, kan vi ikke umiddelbart bruge den i algoritmen; sagen er, at vi er mest interesserede i det "omvendte" tilfælde, hvor det er 0'ernes placering snarere end ikke-nulelementerne, der er udslaggivende. Dette tilfælde har vi formuleret nedenfor; det er helt oplagt, at argumentationen er nøjagtig den samme (grafen i beviset skal blot defineres således at der er en kant mellem en række i og en søjle j netop når $a_{ij} = 0$:

Lad A være en $m \times n$ matrix. Det maximale antal elementer lig nul, som er placeret i forskellige rækker og forskellige søjler, er lig med det minimale antal søjler eller rækker, der skal bruges til at dække alle nulelementer.

Bemærk, at det maximale antal nulelementer, som ikke står i samme række eller søjle, stadig kan identificeres med en maximal parring i en todelt graf. Som vi så i kapitel 9, kan en sådan parring findes ved hjælp af maximal-strøm algoritmen. Vi kan derefter konkludere, at det fundne maximale sæt af nuller i matricen svarer til en minimal dækning med ræker/søjler, noget som viser sig nyttigt.

Kuhn's ungarske metode. Vi vender nu tilbage til assignment-problemet fra før. Vi vil hellere håndtere et minimum end et maximum, og derfor omformuleres problemet lidt: I stedet for matricen

$$R = \begin{pmatrix} r_{11} & \dots & r_{1n} \\ \vdots & & \vdots \\ r_{n1} & \dots & r_{nn} \end{pmatrix}$$

betragter vi en matrix S, hvis (i, j)'te element s_{ij} er givet ved

$$s_{ij} = r - r_{ij}.$$

Vort problem kan da formuleres som: Find en permutation $(i_1, \dots, i_n)$ af tallene $\{1, \dots, n\}$, således at T_S er minimal, når

$$T_S = s_{1i_1} + s_{2i_2} + \dots + s_{ni_n}.$$

Pointen er nu, at den permutation, der minimerer T_S, også vil være minimerende for den matrix, der fremkommer fra S, hvis vi trækker et tal u_i fra alle elementer i

en række i eller trækker v_j fra alle elementer i en søjle j: For enhver permutation af tallene $\{1, \ldots, n\}$ vil vi nemlig formindske det i'te led med u_i, og tilsvarende vil det led, for hvilket permutationen udpeger søjle j, blive formindsket med v_j.

Det betyder, at vi kan omforme den givne matrix S til en matrix på såkaldt *standard form* ved at gennemføre to operationer:

(1) For hver række i erstattes elementerne s_{ij} med

$$s^*_{ij} = s_{ij} - \min_k s_{ik},\ j = 1, \ldots, n;$$

(2) For hver søjle j ersattes s_{ij} med

$$s^*_{ij} = s_{ij} - \min_h s_{hj},\ i = 1, \ldots, n.$$

Den resulterende matrix S^* har nu mindst et nul i hver række og i hver søjle.

Vi bestemmer nu det maximale antal nuller således at hvert 0 står i sin egen række/søjle; lad os kalde det ν. Ifølge korollaret er det lig med det minimale antal rækker eller søjler nødvendige for at dække alle nul-elementer. Der er to tilfælde:

(a) $\nu = n$. Vi kan da finde en permutation af $\{1, \ldots, n\}$ således at der står 0 i alle de pågældende elementer af S^*. Det checkes let, at dette svarer til en løsning til det oprindelige problem.

(b) $\nu < n$. I dette tilfælde må vi fortsætte med en reduceret udgave af problemet: Der er ν rækker eller søjler, således at S^*'s nuller alle ligger i disse rækker eller søjler. Fjern disse; derved fremkommer en matrix $\tilde{S}$, der ikke nødvendigvis er kvadratisk.

Lad

$$h = \min_{i,j} \tilde{s}_{ij},$$

træk h fra alle elementer i S^* som ikke var dækket af de ν rækker eller søjler, og læg h til de elementer, der er dækket to gange. Den resulterende $(n \times n)$ matrix betegnes med S^*_1.

Vi kan nu gentage den oprindelige procedure for S^* på S^*_1; herved fås enten en løsning, eller vi må fortsætte med at revidere problemet som ovenfor, indtil vi har en løsning, for hvilken tilfælde (a) holder.

Det er ikke på forhånd oplagt, at denne procedure faktisk vil ende med en løsning (efter et endeligt antal gennemløb). Men det vil den: Vor revision af problemet svarer til at lægge h til hver af de dækkende rækker/søjler, efterfulgt af subtraktion af h fra hele den resulterende matrix. Med andre ord, vi har

$$\sum_{i,j} (s^*_1)_{ij} = \sum_{i,j} s^*_{ij} - nh(n - \nu) < \sum_{i,j} s^*_{ij},$$

idet vi først til summen af alle elementer i matricen har lagt $hn\nu$ og derefter har trukket hn^2 fra.

Men heraf ser vi, at der i hver iteration sker en reduktion i størrelsen $\sum_{i,j} s^*_{ij}$, og derfor kan processen kun løbe i endelig mange trin.

Eksempel 11.5

Betragt assignment problemet med savings-matrix

$$\begin{pmatrix} 4 & 4 & 5 & 2 & 3 & 8 & 5 \\ 4 & 5 & 3 & 2 & 1 & 9 & 4 \\ 6 & 4 & 1 & 8 & 10 & 2 & 14 \\ 7 & 8 & 5 & 9 & 4 & 12 & 15 \\ 2 & 5 & 3 & 4 & 3 & 5 & 4 \\ 4 & 7 & 2 & 3 & 1 & 6 & 4 \\ 7 & 10 & 10 & 4 & 3 & 4 & 5 \end{pmatrix}$$

Der reduceres først i rækker ved at fratrække minimum, hvorved vi får matricen

$$\begin{pmatrix} 2 & 2 & 3 & 0 & 1 & 6 & 3 \\ 3 & 4 & 2 & 1 & 0 & 8 & 3 \\ 5 & 3 & 0 & 7 & 9 & 1 & 13 \\ 3 & 4 & 1 & 5 & 0 & 8 & 11 \\ 0 & 3 & 1 & 2 & 1 & 3 & 2 \\ 3 & 6 & 1 & 2 & 0 & 5 & 3 \\ 4 & 7 & 7 & 1 & 0 & 1 & 4 \end{pmatrix}$$

Dernæst reduceres i søjler, igen ved at minimum i søjlen fratrækkes alle andre elementer, og vi får:

$$\begin{pmatrix} 2 & 0 & 3 & 0 & 1 & 5 & 1 \\ 3 & 2 & 2 & 1 & 0 & 7 & 1 \\ 5 & 1 & 0 & 7 & 9 & 0 & 11 \\ 3 & 2 & 1 & 5 & 0 & 7 & 9 \\ 0 & 1 & 1 & 2 & 1 & 2 & 0 \\ 3 & 4 & 1 & 2 & 0 & 4 & 1 \\ 4 & 5 & 7 & 1 & 0 & 0 & 2 \end{pmatrix}$$

Vi er nu klar til at lede efter uafhængige nuller. Fra første søjle kan der kun tages ét 0, fra anden søjle ligeledes, og allerede nu har vi valgt rækker, som har de eneste nuller i to senere søjler, så det ser kraftigt ud som om der højest er 5. Det bekræftes af, at alle nuller faktisk kan overdækkes med 5 linier, idet vi bruger 1., 3., 5. og 7. række samt 5. søjle, vist nedenfor med fede typer:

$$\begin{pmatrix} 2 & 0 & 3 & 0 & 1 & 5 & 1 \\ 3 & 2 & 2 & 1 & 0 & 7 & 1 \\ 5 & 1 & 0 & 7 & 9 & 0 & 11 \\ 3 & 2 & 1 & 5 & 0 & 7 & 9 \\ 0 & 1 & 1 & 2 & 1 & 2 & 0 \\ 3 & 4 & 1 & 2 & 0 & 4 & 1 \\ 4 & 5 & 7 & 1 & 0 & 0 & 2 \end{pmatrix}$$

Vi finder nu minimum af de udækkede, som er 1, som trækkes fra alle de udækkede, mens det lægges til de dobbelt dækkede (elementerne i 5.søjle og 1.,3.,5., og 7. række):

Eksempel 11.5, fortsat

$$\begin{pmatrix} 2 & 0 & 3 & 0 & 2 & 6 & 1 \\ 2 & 1 & 1 & 0 & 0 & 7 & 0 \\ 5 & 1 & 0 & 7 & 10 & 1 & 10 \\ 2 & 1 & 0 & 4 & 0 & 7 & 8 \\ 0 & 1 & 1 & 2 & 2 & 3 & 0 \\ 2 & 3 & 0 & 1 & 0 & 4 & 0 \\ 3 & 4 & 6 & 0 & 1 & 0 & 1 \end{pmatrix}$$

Nu gentages vores søgning efter uafhængige nuller, og denne gang lykkes det at finde syv, angivet med fede typer. Dermed har vi fundet en optimale løsning; den fortæller, at 1. søjle skal parres med 5. række, 2. søjle med 1. række, osv.

4. Opgaver

1. Efter en RødeKors indsamling er der indkøbt et antal vareparti, som skal bruges i nødhjælpsarbejdet. Varerne er opmagasineret på flyvestationerne i Karup (100 t), Ålborg (25 t), Værløse (50 t) og Skrydstrup (25 t), og der skal sendes 100 t til Tyrkiet, 50 t til Teheran, 25 t til Etiopien, og 25 t til Bangladesh. Fragtpriserne (kr. pr. kg.) er som følger:

	Tyrkiet	Teheran	Etiopien	Bangladesh
Karup	14	17	12	21
Værløse	13	16	14	18
Ålborg	17	24	10	22
Skrydstrup	12	21	13	20

Find den billigste transportplan.

2. En virksomhed har 4 produktionssteder og 4 forskellige grossister for sit produkt. Virksomheden ønsker at at minimere transportomkostningerne ved leverancer fra fabrik til grossist. Stykomkostningerne ved alternative leveringsmuligheder er som vist:

	grossister:			
	1	2	3	4
fabrikker: 1	2	4	10	9
2	4	1	12	10
3	10	8	2	4
4	9	9	2	2

Der produceres 12 enheder i fabrikkerne 1 og 2, 8 enheder i fabrikkerne 3 og 4. Grossisterne 1 og 2 skal have leveret henholdsvis 9 og 11 enheder, mens 3 og 4 begge skal have 10.

Find den optimale transportplan.

Virksomheden opnår nu rabat på transporten fra fabrik 2 til grossist 3, så at stykprisen på transport ændres fra 12 til 9. Hvorledes påvirker det den optimale plan?

Efterspørgslen fra grossist 1 vokser nu fra 9 til 13. Hvorledes bliver det optimale transportmønster, hvis den øgede efterspørgsel skal klares af fabrik 2?

Det besluttes nu, at det ikke nødvendigvis er fabrik 2, som skal producere de yderligere 4 enheder, men at disse skal laves således, at de samlede transportomkostninger minimeres. Hvordan skal der produceres og transporteres?

3. I forbindelse med en udbygning af infrastrukturen er det besluttet at bygge broer over Langelandsbælt, Smålandshavet, Kattegat og Alssund. Der er indkommet tilbud på alle disse projekter fra 6 forskellige konsortier. Disse tilbud er vist i matricen, hvor rækkerne angiver konsortier, søjler projekter (i ovenfor angivne rækkefølge), alt i mia. 1995-kroner:

$$\begin{pmatrix} 32 & 17 & 29 & 20 \\ 29 & 21 & 25 & 23 \\ 30 & 20 & 25 & 25 \\ 30 & 15 & 27 & 21 \\ 29 & 24 & 23 & 20 \\ 29 & 18 & 29 & 19 \end{pmatrix}$$

Hvem får kontrakterne?

4. En bilforhandler har et lager af brugte biler, og han har haft forhandlinger med et antal købere, hvorved han er nået frem til et bud på, hvad hver kunde vil give for de biler, han har på lager. Denne sammenhæng ser således ud (i tusinde kroner):

	Bil:					
	1	2	3	4	5	6
Køber:						
Antonsen	12	14	9	13	10	16
Børgesen	11	13	15	17	13	11
Corneliussen	9	15	9	14	12	13
Danielsen	10	12	11	13	14	14
Eliassen	13	10	15	10	16	15

Find ud af, hvilke biler der skal afsættes til hvilke kunder.

5. Ved det nyoprettede Institut for Markedsmerkonomer i Ledøje skal man tilrettelægge eksamen. Alle medarbejdere er timelønnede, så der ønskes en arbejdsfordeling, som giver mindst muligt tidsforbrug.

Der er skriftlige eksaminer i fem fag. Antal studenter, der går til eksamen, er anført i parentes:

Markedsanalyse (10)
Erhvervsøkonomi (25)

Afsætningsøkonomi (15)
Nationaløkonomi (30)
Samfundsbeskrivelse (20)

Der er ansat ialt 5 timelærere (A,B,C,D og E). Der er gennemført tidsstudier, således at man har fastlagt tidsforbruget ved retning af en eksamensopgave i hvert fag for hver af lærerne som følger:

Fag:	A	B	C	D	E
Mark.	10	8	12	16	6
Erhv.øk.	6	14	12	10	8
Afs.øk.	6	6	8	8	10
Nat.øk.	8	10	10	8	10
Samf.beskr.	7	9	7	7	9

Lærerne aflønnes i overensstemmelse med disse normerede tidforbrug pr. opgave, og timelønnen er 250 kr. Det er endvidere fastsat, at alle lærere skal have lige mange opgaver til retning.

Find den billigste opgavefordelingsplan.

Det viser sig nu, at til soretering af besvarelserne såvel som addressering og pakning bruges sekretærer på overarbejde. Det foreslås derfor at lade hver lærer tage alle besvarelserne fra netop ét fag.

Find den billigste opgavefordelingsplan i den nye situation.

Efter at opgaverne er pakket i kuverter, kører instituttets betjent ud med dem til lærerne i tjenestebilen. Pakningen af hver kuvert tager forskellig tid for hvert fag, nemlig:

Mark.	Erhv.øk.	Afs.øk.	Nat.øk.	Samf.beskr.
14	20	8	16	10

Betjentens transporttid til hver af timelærerne er: A:10, B:15, C:20, D:13, E:15. Man har et tilbud fra "De sorte cykelryttere" om at gennemføre transporten indenfor 8 minutter (til samme timeløn på 175 kr.). Hvor meget kan spares?

5. Litteratur

Som det fremgår, er transportproblemet blot en særlig udgave af lineær programmering, hvor det er bekvemt at bruge rutiner, der knytter sig til problemet, fremfor en generelle "all-purpose" metode (simplex).

Assignment algoritmen stammer fra Kuhn (1950); når den kaldes "ungarsk", skyldes det anvendelsen af König's sætning.

KAPITEL 12

Optimale rækkefølger

1. Rækkefølgekrav

En lang række af produktionsprocesser vil ved nærmere eftersyn vise sig at bestå af enkeltopgaver, som er indbyrdes afhængige i den forstand, at de lægger beslag på samme faciliteter i form af medarbejdere og produktionsudstyr, og som gensidig stiller krav til hinanden: Det ene job kan ikke påbegyndes, før det andet – eller en kombination af visse andre – job er fuldendt.

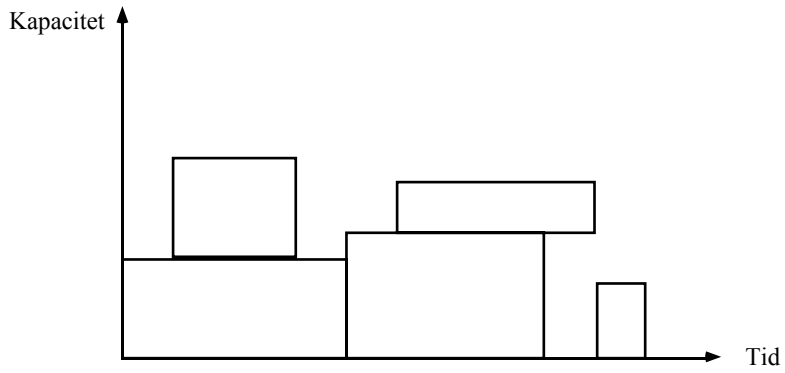
Manglende hensyntagen til sådanne rækkefølgekrav i produktionen vil føre til, at store dele af produktionsapparatet kan komme til at stå stille, mens der ventes på, at en mindre detalje gøres færdig. Det er oplagt utilfredsstillende og inoptimalt, og der er derfor behov for en systematisk metode til at finde den bedst mulige rækkefølge af enkeltdelene af en given produktionsproces, eller et *projekt,* som det gerne kaldes i denne forbindelse.

Der findes en række heuristiske – og i mange tilfælde naturligvis fuldt tilstrækkelige – måder at klare rækkefølgeproblemet på. Den kendteste er nok det såkaldte *Gantt-kort* (se figur 1). Kort fortalt er Gantt-kortet et koordinatsystem med en vandret tidsakse og en lodret akse, som angiver kapaciteten af det aktuelle produktionsapparat. De enkelte jobs i projektet kan da repræsenteres ved rektangulære papstykker og sættes ind i Gantt-kortet. Når det er sket, vil man ved at flytte rundt på jobs kunne finde en rækkefølge, der overholder såvel de indbyggede krav som kapacitetsrestriktionerne.

Om den fundne løsning er den bedst mulige, kan man naturligvis ikke aflæse. Her må man stole på sit øjemål. Det er i mange – måske i hovedparten – af de praktisk forekommende tilfælde også fuldt tilstrækkeligt, men på den anden side giver det ikke hjælp i de tilfælde, hvor der er virkelig mange detaljerede jobs. Hertil kræves en ny systematik, og det er den, vi ser på i kapitlet.

2. Projektet som netværk

Til vor videre behandling af problemstillingen skal vi benytte hjælpemidler fra det foregående, fremfor alt netværk. Ideen er at lade de enkelte jobs i projektet svare til



Figur 1

orienterede kanter i et netværk. Til en sådan kant e har vi da en kapacitet c_e, nemlig *varigheden* af det pågældende job. Rækkefølgekravene kommer til udtryk ved at man for enhver passage i netværket fra s til t må passere kanterne i en bestemt orden.

Inden vi går videre med at opstille netværket hørende til et givet projekt, vil vi illustrere tankegangen med et eksempel, hvor der allerede foreligger et netværk. Betragt således netværket i figur 2. Der er her et projekt bestående af

$$\text{Jobs } j_A,\ j_B,\ \ldots, j_K$$

med visse teknisk givne krav til rækkefølgen, nemlig

- j_A og j_B kommer før j_D,
- j_B kommer før j_E og j_F,
- j_C kommer før J_G,
- j_F og j_G kommer før j_H,
- j_D kommer før j_K.

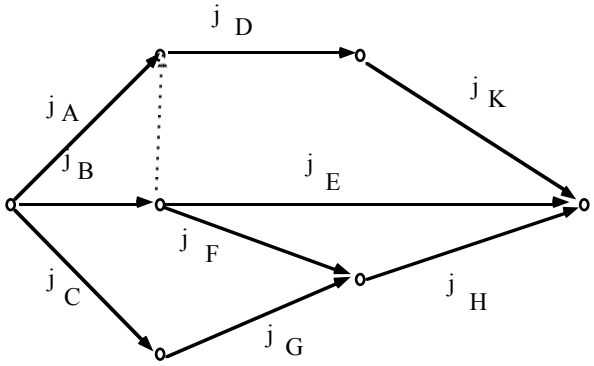
Netværket i figur 2 har netop disse egenskaber. Bemærk at der foruden de kanter, der svarer til jobs i projektet, også er nogle, der ikke modsvares af jobs (såkaldte "dummy"-kanter). De er medtaget for at netværket kan respektere rækkefølgekrav i de situationer, hvor et job afhænger af, at mere end et af de øvrige jobs er færdiggjort, således som det f.eks. gælder i v_1, hvor job j_A *og* job j_B er tilendebragt.

Det er klart nok lettere at checke, at et givet netværk opfylder en række krav, end at opbygge netværket fra angivne rækkefølgebetingelser. Det sidste er dog ikke særlig svært så snart vi har indset, at netværkets punkter ikke har nogen særlig betydning, og at vi kan tilføje dummy-kanter efter ønske.

Den generelle fremgangsmåde er som følger: Lad projektet P bestående af ialt k jobs,

$$P = \{j_1, \ldots, j_k\},$$

og lad der være givet en række betingelser om rækkefølge af jobs, formuleret som en partiel ordningsrelation "kommer før" på P. Vi konstruerer nu et tilhørende netværk



Figur 2

$\mathcal{N}$ med underliggende orienteret graf $\Gamma = (V, E)$ som i en trinvis procedure som følger:

Trin 1: Vi starter med et punkt s og for hvert job j, der er minimal for relationen "kommer før" (hvilket sagt med almindelige ord blot betyder, at der ikke er nogen jobs, der skal komme før), en kant fra s til et nyt punkt $v(j)$. Grafen konstrueret i trin 1 betegnes med Γ_1, og mængden af endnu ikke placerede jobs betegnes med P_1.

Trin $i > 1$**:** Fra mængden P_{i-1} af endnu ikke placerede jobs udtages de jobs j, for hvilke et enkelt job eller en mængde af jobs i Γ_{i-1} kommer før j. Hvis kun et enkelt job kommer før, dvs.

$$j' \text{ kommer før } j,$$

tilføjes en kant fra $v(j')$ til et nyt punkt $v(j)$. Hvis mere end en kant fra Γ_{i-1} kommer før j, lad os sige

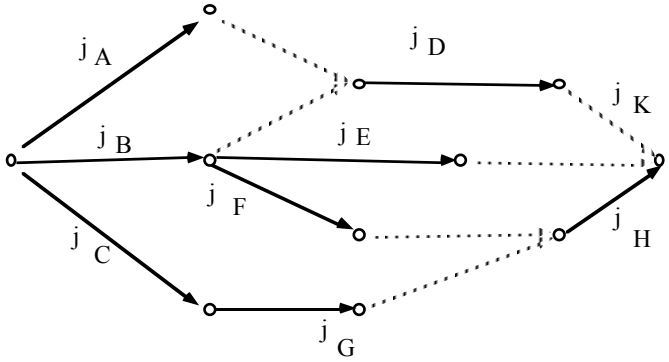
$$j_1, \ldots, j_r \text{ kommer før } j$$

tilføjes for hver kant $j_1, \ldots, j_r$ en dummy-kant til et nyt punkt

$$v(j_1, \ldots, j_r)$$

og en kant fra $v(j_1, \ldots, j_r)$ til endnu et nyt punkt $v(j)$. Den herved fremkomne graf betegnes Γ_i, og mængden af resterende jobs betegnes P_i. Hvis $P_i = \emptyset$, tilføjes der et punkt t og for hvert punkt v i Γ_i, for hvilket der ikke er noget kant, som forlader v, en dummy-kant fra v til t, og proceduren er afsluttet. Hvis $P_i \neq \emptyset$, fortsættes.

Ved at benytte fremgangsmåden angivet her, får man ihvertfald et netværk, der kan bruges. Det er dog ikke eneste mulige repræsentation – og som oftest ikke den mest elegante. Det kan man se af figur 3, hvor vi har brugt proceduren på det samme eksempel som i figur 2. Der kommer lidt rigelig med dummy-kanter ud af det; det



Figur 3

kan der dog ofte rådes bod på efterfølgende, og proceduren ovenfor tjener først og fremmest til at dokumentere, at et repræsenterende netværk kan konstrueres helt mekanisk udfra givne rækkefølgekrav.

3. Den kritiske vej

Når vi har projektet repræsenteret ved et netværk – enten konstrueret som i forrige afsnit eller fremkommet på anden vis – kan vi bestemme, hvor lang tid der ihvertfald må gå, og hvornår vi tidligst kan igangsætte de enkelte jobs. Det sker gennem bestemmelsen af den såkaldte *kritiske vej*.

Fremgangsmåden svarer til, hvad vi foretog os under markeringsprocessen i forbindelse med maximal-strøm algoritmen. I den foreliggende situation er det endda nemmere, idet vi ikke kan have cykler i netværket (det ville give en modstrid med den underliggende ordning af jobs efter om skal efterfølge hinanden). Til gengæld har vi *to* markeringer, en *forlæns markering* og en *baglæns markering*.

> *Forlæns markering:* Her skal alle punkter v markeres med et tal $c(v)$. Der startes med, at s markeres med 0. For et vilkårligt punkt v gælder, at v kan markeres, når alle punkter v', for hvilke der findes en kant $e = (v', v)$, er markeret, og
>
> $$c_f(v) = \max_{v':(v',v)\in E} \{c_f(v') + c_{(v',v)}\}.$$

Det er ikke svært at fortolke de forlæns markeringer: Det er *tidligst mulige start* for alle de jobs, der udgår fra v. Specielt har vi, at $c_f(t)$ er *kortest mulige varighed* af hele projektet.

Hvis projektet skal gennemføres på denne kortest mulige tid, betyder det, at den ventetid, der vil være mellem afslutningen af et job og påbegyndelsen af næste,

vil være mindst mulig. Vi kan gå videre og finde ud af, hvor der er ventetid, og – mere interessant – hvilke jobs der er bestemmende for den samlede varighed af projektet (idet ventetiden mellem disse jobs er 0). Dette sidste giver os den såkaldte *kritiske vej* i projektets netværk: Hvis varigheden skal ned, må der nødvendigvis laves noget om på de jobs, der indgår i den kritiske vej. Omvendt gælder der for de øvrige, ikke-kritiske jobs, at der indenfor en vis grænse kan flyttes rundt med dem, uden at projektet af den grund kommer til at tage længere tid.

Den kritiske vej findes ved at følge den forlæns markering op med en *baglæns markering:*

> *Baglæns markering:* Der startes med t, som markeres med $c_b(t) = c_f(t)$. For et punkt v gælder, at v kan markeres, hvis alle punkter v' med $(v, v') \in E$ er markerede, og
>
> $$c_b(v) = \min_{v':(v,v')\in E} \{c_b(v') - c_{(v,v')}\}.$$

De baglæns markeringer angiver *senest tilladelige sluttidspunkt* for de jobs, der slutter i punktet v. Vi kan nu definere en *kritisk vej* i netværket som en vej fra s til t, for hvilken der for ethvert punkt v på vejen gælder

$$c_f(v) = c_b(v)$$

(denne betingelse svarer netop til, hvad vi nævnte ovenfor om fraværet af ventetid imellem jobs), samt

$$c_{(v,v')} = c_f(v') - c_f(v) = c_b(v') - c_b(v) \tag{1}$$

for alle kanter (v, v') på vejen.

Når vi har den kritiske vej, kan vi gå over til at se på de ikke-kritiske jobs og finde spillerummet for flytning af deres start og sluttidspunkter. Vi definerer den *totale margin* for $e = (v, v')$ (husk at de enkelte jobs er repræsenteret ved kanter i netværket) som

$$m_t[(v, v')] = c_b(v') - c_f(v) - c_{(v,v')},$$

altså som det tidsinterval mellem seneste færdiggørelse af jobbet og tidligste start på det, som bliver til overs når jobbets varighed fraregnes.

Der er således et spillerum på $m_t[(v, v')]$ for placeringen af job $e = (v, v'$. Under visse omstændigheder kan det også have interesse at se på den såkaldte *frie margin,* som er spillerummet i den særlige situation, hvor alle jobs startes så tidligt som muligt. Den kan findes som

$$m_f[(v, v')] = c_f(v') - c_f(v) - c_{(v,v')}.$$

Spillerummet bliver interessant, når vi til de hidtidige overvejelser om hurtigst mulige færdiggørelse tillægger hensynet til at holde omkostningerne lave og at udnytte den givne kapacitet. Muligheden for at lade de enkelte aktiviteter følge efter hinanden snarere end at skulle gennemføre dem samtidig vil typisk være en faktor, som reducerer omkostningerne. Vi skal ikke forfølge dette aspekt nærmere, men blot notere os, at vi med beregningen af den kritiske vej ikke har afsluttet overvejelserne om den bedst mulige rækkefølge, men at det ihvertfald er et væsentligt skridt på vejen.

4. Out-of-kilter metoden til løsning af særlige LP-problemer

Som det var tilfældet med transport og assignment, er der til LP-problemer med særlig struktur udviklet specielle algoritmer, der til deres formål er hurtigere end simplex. Vi skal her se nærmere på en særlig variant til løsning af det såkaldte *minimal omkostning cirkulations* problem, der ser således ud:

$$\min \sum_{i,j} a_{ij} x_{ij}$$

under bibetingelserne

$$\sum_j x_{ji} - \sum_j x_{ij} = 0, \text{ alle } i,$$

$$0 \leq l_{ij} \leq x_{ij} \leq c_{ij}, \text{ alle } i, j.$$

I den oplagte fortolkning knyttet netværk har vi her en strøm, ikke fra en enkelt kilde til en enkelt terminal, men rundt i netværket; x_{ij} angiver, hvad der flyder langs kanten (i, j) fra i til j, og det første sæt bibetingelser er flow conservation: Hvad der ialt går ind i punkt i skal være lig med hvad der ialt går ud af i. Det andet sæt bibetingelser angiver henholdsvis nedre grænse og kapacitet langs kanten (i, j).

Problemet kan gives flere andre fortolkninger (hvilket er en af pointerne ved at se på det i denne sammenhæng), men det gemmer vi til bagefter. Først vil vi se lidt nærmere på, hvorledes man ved en passende skiften mellem det oprindelige, primære, og det duale problem kan komme til en løsning.

Det duale problem har følgende udseende:

$$\max \sum_{i,j} l_{ij} \lambda_{ij} - \sum_{i,j} c_{ij} \gamma_{ij}$$

under bibetingelserne

$$u_j - u_i + \lambda_{ij} - \gamma_{ij} \leq a_{ij}$$

$$\lambda_{ij}, \gamma_{ij} \geq 0.$$

Her er der indført duale variable u_i hørende til det første sæt bibetingelser ovenfor, og variable λ_{ij}, γ_{ij} hørende til henholdsvis nedre grænse og kapacitet langs

kanterne. I det duale problem er der ingen fortegnsrestriktion på u_i'erne, der jo svarer til bibetingelserne givet ved lighedstegn. Det er et godt check af ens forståelse af forholdet primær-dual LP at kontrollere, at det duale problem har denne form.

Vi laver nu en konstruktion, der meget svarer til, hvad der skete ved transport_algoritmen: Vi noterer os, at primært og dualt problem som altid begge har optimale løsninger, hvis blot en af dem har, og at kriterieværdien da er ens. At vi ialt har et sådant sæt af primære og duale løsninger, er ensbetydende med, at følgende betingelser er opfyldte:

$$\begin{aligned} &x_{ij} > 0 \Rightarrow u_j - u_i + \lambda_{ij} - \gamma_{ij} = a_{ij}, \\ &\lambda_{ij} > 0 \Rightarrow x_{ij} = l_{ij} \\ &\gamma_{ij} > 0 \Rightarrow x_{ij} = c_{ij}. \end{aligned}$$

Her kan vi komme af med λ_{ij} og γ_{ij}, idet systemet nedenfor er ækvivalent med det ovenfor:

$$\begin{aligned} &x_{ij} = l_{ij} \Rightarrow u_j - u_i \leq a_{ij} \\ &l_{ij} \leq x_{ij} \leq c_{ij} \Rightarrow u_j - u_i = a_{ij} \\ &x_{ij} = c_{ij} \Rightarrow u_j - u_i \geq a_{ij}. \end{aligned} \tag{1}$$

Lad os efterprøve lidt af det: Antag f.eks. at $x = (x_{ij})$ er en løsning til det primære problem og at der er en kant (i, j) med $0 < l_{ij} = x_{ij} < c_{ij}$. Så har vi

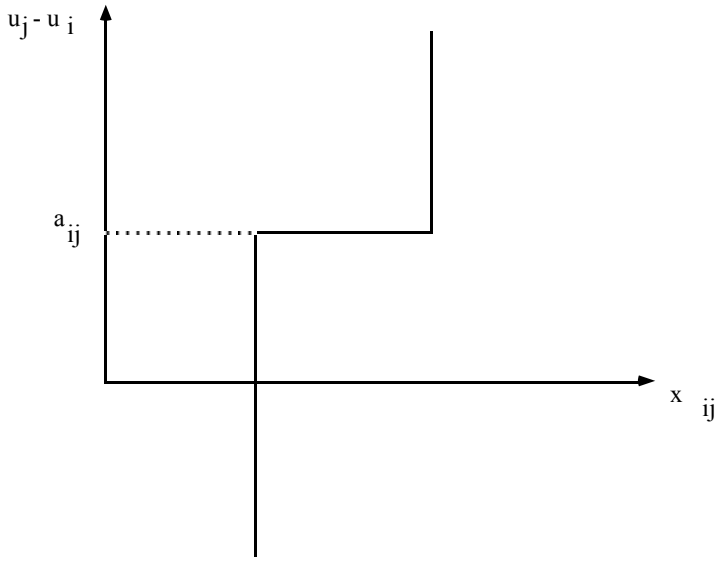
$$x_{ij} > 0 \Rightarrow u_i - u_j + \lambda_{ij} - \gamma_{ij} = a_{ij}.$$

Men da $x_{ij} < C_{ij}$ har vi $\gamma_{ij} = 0$, og da λ_{ij} er ikke-negativ, må der gælde $u_i - u_j \leq a_{ij}$. De øvrige ligninger fås tilsvarende.

Ligningssystemet (1) kaldes kilter-betingelserne, og de kan indtegnes i et såkaldt kilter-diagram som vist i figuren. Punkterne på den knækkede linie er "i kilter" og resten er udenfor. Til et punkt $(x_{ij}, u_j - u_i)$ i diagrammet tilordner vi et kilter-tal $K(x_{ij})$ som er den vandrette afstand fra punktet hen til den knækkede linie. For en ordens skyld skriver vi det lige op: Vi har

$$K(x_{ij}) = \begin{cases} |x_{ij} - l_{ij}| & \text{hvis } u_j - u_i < a_{ij} \\ l_{ij} - x_{ij} & \text{hvis } x_{ij} < l_{ij}, u_j - u_i = a_{ij} \\ x_{ij} - c_{ij} & \text{hvis } x_{ij} < c_{ij}, u_j - u_i = a_{ij} \\ 0 & \text{hvis } l_{ij} \leq x_{ij} \leq c_{ij}, u_j - u_i = a_{ij} \\ |x_{ij} - c_{ij}| & \text{hvis } u_j - u_i > a_{ij}. \end{cases}$$

Formålet med out-of-kilter metoden er at få en cirkulation $x = (x_{ij})$ og et antal punkt variable u_i således at alle kilter betingelser er opfyldt, dvs. så alle kilter tal $K(x_{ij}) = 0$. Man starter med en cirkulation, brugbar eller ej, og et vilkårligt sæt variable u_i. Der skal derefter opdateres; Baggrunden for denne er den såkaldte "farvningssætning" af Minty:



Figur 4

Lad $G = (V, E)$ være en orienteret graf og (s, t) en særlig orienteret kant i G. For enhver farvning af kanterne med farverne grøn, gul og rød gælder ét af de følgende alternativer:

(1) (s, t) er indeholdt i en cykel af gule og grønne kanter, hvor alle gule er orienteret i samme retning,

(2) (s, t) er indeholdt i en cocykel af gule og røde kanter, hvor alle gule kanter har samme retning.

En cocykel i en graf er en minimal mængde af kanter med den egenskab, at når disse kanter fjernes, består den resulterende graf af flere komponenter end den oprindelige graf. Hvis (s, t) er i en cocykel, kan vi opdele den oprindelige grafs punkter i to mængder S og T, så $s \in T$, $t \in T$, og kanterne forbindende S og T netop er dem fra cocyklen. Vi skal ikke give et bevis for denne grafteoretiske sætning; det er dog ikke er særlig vanskeligt.

Nok så væsentligt her er anvendelsen af farvningssætningen. Her er for det første vor farvning af kanter:

(a) Farv (i, j) grøn hvis den er i kilter og det er muligt både at øge og mindske x_{ij} uden at den ryger ud. For en sådan kant har vi

$$l_{ij} < x_{ij} < c_{ij} \text{ og } u_j - u_i = a_{ij}.$$

(b) Farv (i, j) gul hvis det er muligt at øge x_{ij}, men ikke at mindske den, uden at kilter-tallet forøges. Det vil sige, at vi må have enten

$$x_{ij} < c_{ij} \text{ og } u_j - u_i > a_{ij}$$

eller

$$x_{ij} \leq l_{ij} \text{ og } u_j - u_i = a_{ij}$$

eller

$$x_{ij} < l_{ij} \text{ og } u_j - u_i < a_{ij}.$$

(c) Farv en kant (i, j) gul *og skift dens orientering* hvis det er muligt at mindske x_{ij}, men ikke øge den, uden af forøge kilter-tallet. For sådanne kanter har vi enten

$$x_{ij} > c_{ij} \text{ og } u_j - u_i > a_{ij}$$

eller

$$x_{ij} \geq c_{ij} \text{ og } u_j - u_i = a_{ij}$$

eller

$$x_{ij} > l_{ij} \text{ og } u_j - u_i < a_{ij}.$$

(d) Farv (i, j) rød hvis x_{ij} hverken kan øges eller mindskes uden forøgelse af kilter-tallet, dvs.

$$x_{ij} = c_{ij} \text{ og } u_j - u_i > a_{ij}$$

eller

$$x_{ij} = l_{ij} \text{ og } u_j - u_i < a_{ij}$$

Vi har nu udtømt alle muligheder. Bemærk, at såvel røde som grønne kanter er i kilter. De gule er det kun i specielle tilfælde, nemlig hvis $(x_{ij}, u_j - u_i)$ er et knækpunkt på linien i kilterdiagrammet.

Betragt nu en kant (t, s) som ikke er i kilter, og brug farvningssætningen. Antag, at der er en gul-grøn cykel C, hvor alle gule kanter er i samme retning som (s, t). Vend nu alle de kanter, som fik skiftet orientering under farvningen. Hvis vi øger strømmen gennem (t, s) med et $\delta > 0$, så vil vi reducere kilter-tallet med dette δ. Hvis (t, s) selv er en af de kanter, hvis orientering blev vendt om, betyder det, at vi reducerer strømmen langs (s, t) med δ. En forøgelse af strømmen i kanterne orienteret som (t, s) i cyklen C og en reduktion i kanterne orienteret (s, t) vil ikke øge kiltertallet i nogen kant, og det kan formindske kiltertallet i nogle kanter. Altså er $C \setminus (t, s)$ en forbedrende vej fra s til t.

Når vi har en gul-grøn cykel, skal vi bestemme det størst mulige δ i argumentet ovenfor. Det gør vi som følger: Lad C_{gul} og C_{gr} være de gule og grønne kanter i cyklen C, og sæt toptegnene $+$ og $-$ på hvis strømmene i kanterne skal henholdsvis forøges eller formindskes med δ. Vi har da, at ingen kant, som er i kilter, vil ryge ud, så længe δ ikke overstiger den mindste af

$$\delta_1 = \min\{c_{ij} - x_{ij} | (i, j) \in C^+_{gul} \cup C^+_{gr}, u_j - u_i = a_{ij}\},$$

$$\delta_2 = \min\{x_{ij} - l_{ij} | (i, j) \in C^-_{gul} \cup C^-_{gr}, u_j - u_i = a_{ij}\}.$$

Forøgelsen δ vil holde sig under, hvad der er nødvendigt for at bringe en udenfor-kilter kant ind, hvis den holdes under det mindste af

$$\delta_3 = \min\{|c_{ij} - x_{ij}| \, | (i, j) \in C^+_{gul} \cup C^-_{gul}, u_j - u_i > a_{ij}\},$$

$$\delta_4 = \min\{|x_{ij} - l_{ij}| \mid (i,j) \in C^+_{gul} \cup C^-_{gul}, u_j - u_i < a_{ij}\}.$$

Vi vælger derfor $\delta = \min\{\delta_1, \delta_2, \delta_3, \delta_4\}$ og opdaterer. Dette fungerer med mindre δ er ubegrænset; i det tilfælde er alle δ_i fremkommet ved at minimere over en tom mængde, og der er ikke nogen endelig optimal cirkulation.

Antag dernæst, at der er en gul-rød cocykel C, som vi i overensstemmelse med betragningerne ovenfor kan skrive (S,T) med $s \in S$, $t \in T$; alle de gule kanter er orienteret i samme retning som (t,s). Vi vender nu orienteringen (tilbage) til de kanter, der fik den vendt under farvningen. En lille forøgelse $\varepsilon > 0$ i u_i'erne hørende til alle punkter i T vil kun påvirke forskellen $u_j - u_i$ for de kanter, der er i cocyklen. Videre gælder der, at den ikke vil øge kilter-tallet for nogen kant, og den kan reducere det for visse kanter.

På samme måde som tidligere indfører vi notation C_{gul} og C_r for de gule og røde kanter i cocyklen C (her opfattet som en mængde af kanter), og vi sætter $+$ og $-$ på delmængderne af kanter (i,j), for hvilke $u_j - u_i$ skal henholdsvis forøges og formindskes. På samme måde som tidligere kan vi lægge grænser på ε: Ingen kant som er i kilter, ryger ud, så længe ε holdes under

$$\varepsilon_1 = \min\{u_j - u_i - a_{ij} | (i,j) \in C^-_r, x_{ij} = c_{ij}\},$$

$$\varepsilon_2 = \min\{a_{ij} - u_j + u_i | (i,j) \in C^+_r, x_{ij} = l_{ij}\}.$$

Forøgelsen ε vil ikke overstige, hvad der er nødvendigt for at komme i kilter, hvis man ikke er det, såfremt den er mindre end

$$\varepsilon_3 = \min\{u_j - u_i - a_{ij} | (i,j) \in C^-_{gul}, l_{ij} \le x_{ij} < c_{ij}\},$$

$$\varepsilon_4 = \min\{a_{ij} - u_j + u_i | (i,j) \in C^+_{gul}, l_{ij} < x_{ij} \le c_{ij}\}.$$

Ialt vælges derfor $\varepsilon = \min\{\varepsilon_1, \varepsilon_2, \varepsilon_3, \varepsilon_4\}$. Der er nu tre muligheder:

(1) ε er ubegrænset, således at hver af $\epsilon_i$erne er minima over tomme mængder. Dette kan kun forekomme, hvis $x_{ij} > c_{ij}$ for alle kanter fra S til T og $x_{ij} \le l_{ij}$ for alle kanter fra T til S, samt $x_{is} < l_{is}$. Da nettostrømmen fra S til T er nul (flow conservation sammen med det faktum, at (S,T) er en klassedeling af grafens punkter), får vi

$$\sum_{i \in S, j \in T} l_{ij} > \sum_{i \in T, j \in S} c_{ij}.$$

Men heraf følger, at der ikke kan eksistere nogen cirkulation.

(2) ε er endelig og $= \varepsilon_3$ eller ε_4. Så er mindst én kant udenfor kilter kommet ind; ingen kilter-tal er øget, og mindst ét er formindsket.

(3) ε er endelig og mindre end såvel ε_3 som ε_4. Ingen kilter-tal er forøgede, og nogle kan være formindskede. Der er mindst en kant, som var rød men bliver gul ved næste farvning. En sådan kant opfylder

$$l_{ij} = x_{ij} < c_{ij} \text{ for } i \in S, j \in T,$$
$$l_{ij} < x_{ij} = c_{ij} \text{ for } i \in T, j \in S.$$

Der kan også være kanter, der skifter farve fra gul til rød. For sådanne kanter har vi

$$l_{ij} < x_{ij} = c_{ij} \text{ for } i \in S, j \in T,$$
$$l_{ij} = x_{ij} < c_{ij} \text{ for } i \in T, j \in S,$$

Dermed er vi færdige med at beskrive algoritmen, der stopper, når alle kilter-tal er nul, eller det fra et trin i algoritmen fremgår enten at problemet er ubegrænset (ikke har noget endeligt minimum) eller at det ikke har nogen brugbar løsning. Vi mangler at redegøre for, at proceduren nødvendigvis må stoppe; her vil vi antage, at alle data i problemet er heltallige, og at vi starter med en heltallig cirkulation.

Hver gang vi finder en gul-grøn cykel, reduceres kiltertallet med mindst δ, og fra heltalligheden får vi, at δ er mindst 1. Hvis K er kilter-tallet for start-cirkulationen, skal vi altså ikke finde gul-grønne cykler mere end K gange.

Hver gang vi finder en gul-rød cocykel (og problemet ikke afsløres uløseligt som i tilfæde (1) ovenfor), får vi enten en kant udenfor kilter ind (tilfælde (2)), hvorved vi igen reducerer med mindst 1, eller (tilfælde (3)) vi får skiftet en rød kant til gul. Dette sidste kan imidlertid ikke ske mere end $n-1$ gange i træk, hvor n er antallet af punkter i netværket:

Antag at samme kant (t, s) bruges til anvendelse af farvningssætningen indtil vi får en gul-grøn cykel. Hver gang en cocykel opdages og vi er i tilfælde (3) , skifter en rød kant farve til gul på en sådan måde, at endnu et punkt i T kan nås fra s næste gang man finde cocykler; alle de punkter, der tidligere kunne nås, kan stadig nås. Der kan derfor højest gennemføres $n-1$ sådanne runder.

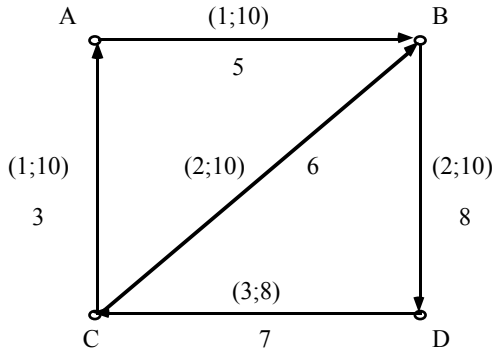
Vi har nu, at kilter-tal reduceres pånær i sådanne runder, hvor der opdages en cocykel og vi er i tilfælde (3). Da dette kun kan gentage sig et vist antal gange, når det forekommer, må algoritmen ende med kilter-summen 0.

Eksempel 12.1

For at illustrere out-of-kilter algoritmens funktion i en tilpas simpel situation betragter vi i fortsættelsen følgende tilfælde: Der er givet et rutenet mellem fire pladser i den indre by, A, B, C og D, og opgaven er at finde en busbetjening (målt ved antal busser i timen). Uheldigvis er der mange ensrettede veje, og man baserer sig derfor på ringlinier. Spørgsmålet er nu, hvor mange busser man kan lade cirkulere i rutenettet, hvis det overordnede mål er at minimere omkostningerne.

Lad os antage, at rutenettet er som vist på grafen nedenfor: Tallene i parentes ved hver kant (svarende til gader, der alle er ensrettede) angiver nedre begrænsning (man har en forpligtelse til at opretholde denne betjening) og kapacitet (der kan ikke sendes flere igennem uden trafikpropper). Det sidste tal viser omkostninger pr. bus,

Eksempel 12.1, fortsat



Figur 5

der sendes igennem denne gade. På tabelform ser det således ud: Omkostningerne er givet ved tabellen

·	5	·	·
·	·	·	8
3	6	·	·
·	·	7	·

mens nedre og øvre begrænsninger fremgår af den tilsvarende tabel

·	$(1;10)$	·	·
·	·	·	$(2;10)$
$(1;10)$	$(2;10)$	·	·
·	·	$(3;8)$	·

Vi vil bruge out-of-kilter algoritmen til at finde den billigste cirkulation. Det første trin består i at finde en eller anden startløsning, der omfatter såvel strømmene x_{ij} som punkt-variablene u_i. Der kræves ikke, at strømmene skal overholde øvre og nedre begrænsning, men flow conservation skal være opfyldt. En nem måde at sikre dette er at vælge alle til 0. For punkt-variablenes vedkommende er det tilsvarende det mest oplagte at starte med 0. Vi får dermed den følgende ret banale startløsning:

·	0	·	·
·	·	·	0
0	0	·	·
·	·	0	·

0
0
0
0

Vi går nu i gang med at farve kanterne. Det viser sig hurtigt at være ganske nemt: Tag f.eks. kanten fra A til B; da den aktuelle strøm er mindre end minimum, er kanten gul. Nøjagtig samme argument holder for alle de andre kanter.

Vi skal nu til at lede efter cykler/cocykler. Først vælger vi en gul kant; det er ligegyldigt hvilken, lad os tage (C, B). Vi forsøger nu at få denne kant til at indgå i en grøn-gul cykel, hvor alle gule kanter (og det vil her sige samtlige) er orienteret på samme måde. Der er et oplagt valg her, nemlig cyklen $((C, B), (B, D), (D, C))$.

Eksempel 12.1, fortsat

Nu kan vi opdatere strømmene med det mindste tal, som får strømmen i en eller anden gul kant til at overholde bibetingelserne. Det er i dette tilfælde 2. Den nye løsning ser herefter således ud:

·	0	·	·
·	·	·	2
0	2	·	·
·	·	2	·

0
0
0
0

Man kan gøre en prøve gående ud på at i'te rækkesum skal være lig i'te søjlesum; det er netop flow conservation.

Vi er klar til næste skridt. Først farves der; kanten (A, B) er gul som før, og det samme gælder kanten (C, A); aktuel strøm ligger under nedre grænse. Kanterne (C, B) og (B, D) er begge røde, fordi strøm er lig nedre grænse, mens forskel i punkt-variabel i endepunkterne (som er nul for begge kanter) er mindre end omkostningskoefficient. Kan (D, C) er stadig gul. Ialt har vi altså:

·	gul	·	·
·	·	·	rød
gul	rød	·	·
·	·	gul	·

Vi er klar til at søge efter cykel/cocykel: Vælg en vilkårlig gul kant, f.eks. kanten (C, A) Det er ret let at se, at der ikke er nogen gul-grøn cykel, for når vi kører fra C til A og videre til B, er der ikke nogen gul eller grøn kant, der fører videre. Altså må der være en rød-gul cocykel indeholdende (C, A). Det svarer til at alle punkterne skal deles i to mængder S og T, hvor C er i S og A er i T. Alle gule kanter mellem S og T skal være orienteret på samme måde, og det giver os cocyklen med det samme: A og B må høre sammen i T, og C og D må så ligge i S (check det selv, deler vi punkterne på anden måde, kommer der til at gå en gul kant i forkert retning mellem mængderne).

Så skal vi opdatere, og når det er en cocykel, er det punktvariablene, der skal tilpasses. Det gør vi ved at øge u_i'erne i T så meget, at en af cocyklens kanter får forskel i punktvariable til at være lig omkostning. Vi ser, at det bliver den røde kant (C, B), der bestemmer den mindste af disse forskelle, som er 6 $(= 6 - 0)$. Vi får dermed opdateret løsning:

·	0	·	·
·	·	·	2
0	2	·	·
·	·	2	·

6
6
0
0

Der er jo ikke sket så meget denne gang, for vi har kun ændret på punktvariablene, der alligevel er hjælpestørrelser. Det går dog alligevel fremad, som vi vil se om lidt.

Vi er klar til tredie runde og farver: Kanten (A, B) har $(x_{AB}, u_B - u_A) = (0, 0)$ og er gul. Kanten (C, A) har tilsvarende koordinater $(0, 6)$ og er også gul. Også (C, B) er gul med koordinater $(2, 6)$, der ligger lige i kilter-liniens knæk. Kanten (B, D) har koordinater $(2, -6)$ og er rød, og endelig er kanten (D, C) med $(2, 0)$ gul.

Eksempel 12.1, fortsat

Vi vælger en gul kant, f.eks. (A, B), men må hurtigt opgive at finde en cykel, for vi kan ikke komme videre fra B. Altså er der en cocykel, og den må skille punkterne A, B, C fra D; ellers ville en gul kant være orienteret forkert. Derefter kan der opdateres; vi kan bringe den røde kant (B, D) op på den vandrette del af kilterlinien ved at lægge 14 til u_D. Den nye løsning bliver så:

·	0	·	·	6
·	·	·	2	6
0	2	·	·	0
·	·	2	·	14

Så farver vi igen. Man checker let at (A, B),(C, A) og (D, C) stadig er gule, at (B, D) er blevet gul, og at endelig (C, B) er blevet gul. Vælg en gul kant, f.eks. (A, B) (man må gerne vælge samme kant i flere forskellige omgange); der er en gul cyke, nemlig hele vejen rundt A, B, D, C, A. Når det er en cykel, opdaterer vi strømme, så vi får en kant ind på kilterlinien. Vi ser, at (B, D) så kommer på kilterliniens vandrette stykke, men det gør jo ikke noget. Vi opdaterer med 1 og får ny løsning:

·	1	·	·	6
·	·	·	3	6
1	2	·	·	0
·	·	3	·	14

Nok en farvning: Alle kanter er gule pånær (B, D), der er blevet grøn. Men ved nærmere eftersyn ser vi, at alle kanterne faktisk er i kilter; de fleste ligger på den nederste lodrette del af kilter-linien, og (B, D) på den vandrette. Men så har vi en optimal løsning, og vi er færdige.

5. Anvendelse af out-of-kilter metoden til project scheduling

Metoden, som blev udviklet i forrige afsnit, fremstod som designet specielt til cirkulationsproblemer. Dem er der umiddelbart ikke så mange af, så hvis den ikke kunne andet, var der ikke meget grund til at bruge tid på den. Det kan den imidlertid.

Optimal kapacitetsforøgelse i netværk: Vi har tidligere set på, hvorledes man finder en maximal strøm i et givet netværk. Det er imidlertid i mange sammenhænge interessant at lave om på netværkets kapaciteter, og hvis det er muligt, kan metoden fra kapitel 7 ikke bruges længere. Antag, at det er muligt at øge på kapaciteten c_{ij} i kanten fra i til j, men at der er en omkostning på a_{ij} forbundet med hver enheds kapacitetsforøgelse. Hvis den faktiske forøgelse er y_{ij}, kan vi opskrive problemet om at tilpasse kapaciteter med minimale omkostninger, således at den samlede strøm bliver v; det får formen

$$\min \sum_{i,j} a_{ij} y_{ij}$$

under bibetingelserne

$$\sum_j x_{ji} - \sum_i x_{ij} = \begin{cases} -v & \text{for } i = s \\ 0 & \text{for } i \neq s, t \\ v & \text{for } i = s \end{cases}$$

$$0 \leq x_{ij} \leq c_{ij} + y_{ij},$$

hvor x_{ij} er strømmen langs kant (i, j).

Umiddelbart er dette ikke et cirkulationsproblem, men det kan det nu ret nemt blive, nemlig ved tilføjelsen af en enkelt (kunstig) kant fra t til s med $l_{ts} = c_{ts} = v$ (således at der altså nødvendigvis skal gå v igennem fra t til s). Så er vi stort set tilbage i problemet fra forrrige afsnit, bortset fra, at vi har lidt for mange variable (nemlig både y_{ij} og x_{ij}. Nu gælder der jo at $y_{ij} = x_{ij} - c_{ij}$ i alle de tilfælde, hvor $x_{ij} \geq c_{ij}$ (der er jo ingen mening i at øge kapaciteten mere end nødvendigt), og 0 ellers, så problemet kan omformuleres til minimering af $\sum_{ij} \bar{a}_{ij}(x_{ij})$, hvor $\bar{a}_{ij}$ er funktionen

$$\bar{a}_{ij}(x_{ij}) = \begin{cases} a_{ij}(x_{ij} - c_{ij}) & \text{hvis } x_{ij} \geq c_{ij} \\ 0 & \text{ellers} \end{cases}$$

Denne funktion er uheldigvis ikke lineær, men heldigvis kan out-of-kilter metoden stadigvæk anvendes, når omkostningerne i hver kant som her er en konveks og stykkevis lineær funktion af strømmen (kilterdiagrammet får flere trappetrin).

Optimering i projekt-netværk: Et andet anvendelsesområde har at gøre med kapitlets emne : Vi betragter et projekt-netværk, hvor vi til hvert punkt i af netværket har defineret en variabel u_i, som angiver tidspunktet for, hvornår den begivenhed, der svarer til punktet, indtræffer. Der er en fast øvre grænse T for, hvornår det hele skal være afsluttet, og for hvert job – svarende til kant (i, j) i projekt-netværket – er der angivet såvel en "normal" varighed a_{ij} som en mindste tænkelige varighed b_{ij}. Der er givet omkostninger c_{ij} ved forkortelse af den normale varighed i kanten (i, j). Det ligger i problemet, at der bliver behov for at forkorte varigheden i nogle eller alle kanter; den faktiske varighed af jobbet betegnes t_{ij}.

Det generelle omkostningsminimeringsproblem kommer da til at gå ud på at minimere

$$\sum_{i,j} c_{ij}(a_{ij} - t_{ij}),$$

den samlede omkostning ved at gennemføre jobbet, eller ækvivalent (da a_{ij}'erne er konstanter, at maximere

$$\sum_{ij} c_{ij} t_{ij}$$

under bibetingelserne, som er

$$\begin{aligned} u_i + u_j &\leq T \\ u_i - u_j + t_{ij} &\leq 0 \\ b_{ij} \leq t_{ij} &\leq a_{ij} \end{aligned}$$

for alle (i, j). Bemærk, at der ikke er yderligere fortegnsrestriktioner på u_i'er eller t_{ij}'er.

Det duale til dette problem er at minimere

$$\sum_{ij} a_{ij}\alpha_{ij} - \sum_{i,j} b_{ij}\beta_{ij} + Tv$$

under bibetingelserne

$$\sum_j x_{ji} - \sum_j x_{ij} = \begin{cases} -v & \text{if } i = s \\ 0 & \text{if } i \neq s, t \\ v & \text{if } i = t \end{cases}$$

$$x_{ij} + \alpha_{ij} - \beta_{ij} = c_{ij}$$

$$x_{ij}, \alpha_{ij}, \beta_{ij} \geq 0.$$

Her omformes lidt: Da en optimal løsning må være sådan, at $x_{ij} \leq c_{ij}$ implicerer $\alpha_{ij} = c_{ij} - x_{ij}$ og $\beta_{ij} = 0$ (det er den komplementære slackness), og tilsvarende $x_{ij} > c_{ij}$ medfører $\alpha_{ij} = 0, \beta_{ij} = x_{ij} - c_{ij}$, kan vi nøjes med at se på problemet

$$\min \sum_{ij} T_{ij}(x_{ij}) + Tv$$

under bibetingelserne

$$\sum_j x_{ji} - \sum_i x_{ij} = \begin{cases} -v & \text{for } i = s \\ 0 & \text{for } i \neq s, t \\ v & \text{for } i = t \end{cases}$$

$$x_{ij} \geq 0$$

hvor T_{ij} er funktionen givet ved

$$T_{ij} = \begin{cases} -a_{ij}x_{ij} & \text{for } x_{ij} \leq c_{ij} \\ -a_{ij}c_{ij} - b_{ij}x_{ij} & \text{for } x_{ij} \geq c_{ij} \end{cases}$$

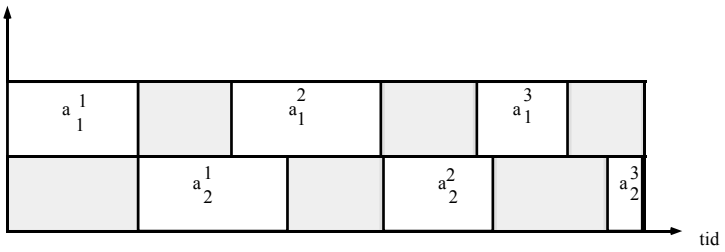
Som tidligere kan vi få dette til at blive cirkulationsproblem ved at tilføje en kant fra t til s (restriktioner ikke nødvendige). Omkostningen langs kanter er igen stykkevis lineær og konveks, så out-of-kilter metoden kan anvendes.

6. Flow-shop problemet

I det foregående afsnit har vi diskuteret problemet om at finde den hurtigst mulige færdiggørelse af et projekt bestående af flere enkelte aktiviteter, givet at der – bortset fra rækkefølgekravet – ikke var noget begrænsning på, hvor mange af disse aktiviteter der kunne udføres på samme tid.

I praksis vil der imidlertid ofte være sådanne begrænsninger; en måde, hvorpå de kan opstå, er at hver enkelt aktivitet består af flere processer, *jobs,* der hver især lægger fuldt beslag på en bestemt maskine, så længe de varer. Vi skal se på denne situation i en forholdsvis simpel udgave: Hver aktivitet består af 2 jobs, der kræver hver sin maskine, og der er kun én af hver slags. Til gengæld er de enkelte aktiviteter uafhængige af hinanden. Vi har indført en nummerering således at job 1 af hver aktivitet foretages på maskine 1, job 2 på maskine 2.

Generelt taler man om en *flow-shop* når aktiviteternes enkelte jobs er ordnet således at efterfølgende jobs kræver maskiner med højere nummer, mens aktiviteterne selv kan placeres i vilkårlig orden. Vi skal dog kun se på det ovennævnte tilfælde med 2 jobs og 2 maskiner, og det er der (som vi senere skal komme ind på) særlige grunde til. Situationen kan illustreres i Figur 6, hvor vi har tegnet et Gantt-kort for de to maskiner. Hver enkelt aktivitet består at et job på den øverste maskine, der skal være færdigt før dets job på den nederste starter.



Det grundlæggende problem er som før at finde en den måde at lade de enkelte aktiviteter udføre på de to maskiner, som minimerer den samlede tid fra begyndelse til slut. Kaldes de varigheden af de enkelte aktiviteter i for

$$a^i = a_1^i + a_2^i,$$

hvor vi har skrevet den samlede varighed som summen af de enkelte jobs varighed, skal vi finde en måde at ordne de enkelte jobs på maskine 1 og de enkelte jobs må maskine 2, således at den samlede sum af jobs og ventetider bliver mindst mulig.

Formelt set er en ordning af aktiviteternes passage gennem en maskine j en *permutation* af aktiviteterne, dvs. en bijektiv afbildning σ fra indexmængden $I_A = \{1, \ldots, m\}$ for maskiner til sig selv, således at $\sigma(i)$ angiver den aktivitet som passerer gennem maskinen som nummer i i rækkefølgen.

Først noterer vi os følgende:

Hvis σ_1 og σ_2 er en løsning til problemet om minimering af passagetid, da kan vi antage $\sigma^1 = \sigma^2$.

Med andre ord, hvis ikke aktiviteterne passerer igennem maskinerne i samme rækkefølge, da kan vi ordne om på rækkefølgen i den ene, så at de bliver ens, uden at dette forøger passagetiden.

Bevis for vort hjælperesultat går som følger: Lad i_1 være den første maskine i rækkefølgen på maskine 1, hvor der ikke er overensstemmelse med de to rækkefølger, dvs.

$$i_1 = \min\left\{\sigma_1^{-1}(i) \neq \sigma_2^{-1}(i)\right\};$$

sæt $\sigma_2^{-1}(i_1) = i_2$. Da ved vi, at aktiviteten i_2 ikke kan påbegyndes på maskine 2, før den er behandlet på maskine 1, altså ihvertfald ikke før aktivitet i_1 er tilendebragt på maskine 1. Vi kan nu flytte aktivitet i_1 på maskine 2 op umiddelbart foran aktivitet i_2 og rykke starttidspunkterne for disse aktiviteter tilbage med $a_2^{i_1}$. Vi ved da, at alle de aktiviteter, der skal være færdige på maskine 1 før de starter på maskine 2, stadig er det (for vi har flyttet alle til senere tidspunkt på maskine 2 eller holdt dem uændret, pånær aktivitet i_1, som kunne startes på det nye tidspunkt. Endelig har vi på grund af konstruktionen, at sluttidspunktet for maskine 2 ikke er ændret.

Fortsættes på samme måde, fås i endelig mange trin to identiske jobrækkefølger. □

Vi har således at vi kan nøjes med at kontrollere rækkefølger, som er ens for de to maskiner. Vi skal se lidt nærmere på en simpel procedure til at forbedre samlet gennemløbstid.

Betragt en vilkårlig rækkefølge σ; for hver maskine vil der blive arbejdet med aktivitet $\sigma(1), \sigma(2), \ldots, \sigma(m)$, og der vil være noget spildtid før og efter. Det er oplagt, at vi i vor undersøgelse kan antage, at al spildtid på maskine 1 ligger i slutningen: Der startes umiddelbart med $\sigma(1)$, der fortsættes umiddelbart efter med $\sigma(2)$ osv. Derimod vil der være spildtid $x_{\sigma(1)}$ ved maskine 2 inden første aktivitet kan passere, og derefter eventuelt mellem de enkelte passager. Der bliver til gengæld ingen spildtid i slutningen for maskine 2, for da er hele projektet jo slut.

Det følger heraf, at vi kan skrive den samlede gennemløbstid som

$$T_\sigma = x_2^{\sigma(1)} + a_2^{\sigma(1)} + \cdots + x_2^{\sigma(m)} + a_2^{\sigma(m)},$$

og vor opgave er at vælge permutationen σ således at denne sum bliver mindst mulig.

Vi kan umiddelbart give et bud på, hvad denne gennemløbstid T_σ *mindst* må være: Vi kan ikke slutte projektet før alle aktiviteter har passeret maskine 1 *og* den sidste har passeret maskine 2, dvs.

$$T_\sigma \geq (a_1^{\sigma(1)} + \cdots + a_1^{\sigma(m)}) + a_2^{\sigma(m)}; \tag{2}$$

tilsvarende kan vi ved at se på projektets start, hvor første aktivitet skal igennem maskine 1 før arbejdet kan begynde på maskine 2, få at der gælder

$$T_\sigma \geq a_1^{\sigma(1)} + (a_2^{\sigma(1)} + \cdots + a_2^{\sigma(m)}). \tag{3}$$

Dermed har vi også et bud på en procedure til at ordne aktiviteterne: Ønsker vi den samlede tid T_σ mindst mulig, må vi søge at få de nedre begrænsninger, som ligningerne (2) og (3) giver os, minimale. Det ses, at de kun afhænger af, hvad der står først, henholdsvis sidst; summerne er jo uafhænginge af ordningen. Vi kan da undersøge aktiviterne og finde en, således en aktivitet $a^i = (a_1^i, a_2^i)$, således at

$$\min\left\{a_1^i, a_2^i\right\}$$

er minimal. Er dette minimum antaget af a_1^i, sætter vi $\sigma(i) = 1$ (dvs. aktivitet i kommer først), ellers sættes $\sigma(i) = m$ (aktivitet i kommer sidst. Der er da kun $m - 1$ aktiviteter tilbage, og der gennemføres samme procedure med disse.

Denne procedure kaldes *Johnson's algoritme;* vi skal undersøge den lidt nærmere for at vise, at den faktisk giver os en rækkefølge σ, som minimerer T_σ. Først bemærker vi følgende:

Antag at aktiviteterne i og j opfylder

$$\min\left\{a_1^i, a_2^j\right\} < \min\left\{a_1^j, a_2^i\right\}.$$

Da vil der ved ordning efter Johnson's algoritme gælde, at

$$\sigma(i) < \sigma(j).$$

Antag nemlig at vi efter et antal skridt i algoritmen har en mængde af aktiviteter, der indeholder både i og j. Der er to tilfælde:
(1) $a_1^i \leq a_2^j$: I så fald er $a_1^i < a_2^i$, og

$$\min\left\{a_1^i, a_2^i\right\} < \min\left\{a_1^j, a_2^j\right\},$$

og aktivitet j vil aldrig blive udtaget, så længe aktivitet i er til stede. Når i bliver udtaget, placeres den først.
(2) $a_2^j < a_1^i$: I dette tilfælde er

$$\min\left\{a_1^j, a_2^j\right\} < \min\left\{a_1^i, a_2^i\right\},$$

og $a_1^j > a_2^j$, så i vil aldrig blive udtaget, når aktivitet j er til stede; aktivitet j placeres sidst, når den udtages.

I begge tilfælde ses det, at $\sigma(i) < \sigma(j)$. □

Det ses let, at der omvendt må gælde

$$\min\left\{a_1^i, a_2^j\right\} \leq \min\left\{a_1^j, a_2^i\right\} \quad (4)$$

når aktiviteterne er ordnet i overensstemmelse med Johnson's algoritme.

For at sikre os, at vi er kommet frem til den bedste rækkefølge, bruger vi følgende:

Antag at der gælder $\sigma(i) < \sigma(j)$ netop når

$$\min\left\{a_1^i, a_2^j\right\} \leq \min\left\{a_1^j, a_2^i\right\}. \quad (4)$$

Da vil den samlede gennemløbstid T_σ ikke vokse ved en ombytning af to aktiviteter i rækkefølgen.

Først bemærkes, at det afgørende led i summen, som definerer $T\sigma$, nemlig

$$x^{\sigma(1)} + \cdots + x^{\sigma(m)}$$

kan skrives som

$$\sum_{i=1}^{m} x^{\sigma(i)} = \max\{y_1, \ldots, y_m\},$$

hvor

$$y_i = \sum_{j=1}^{i} a_1^{\sigma(j)} - \sum_{j=1}^{i-1} a_2^{\sigma(j)}. \quad (5)$$

Dette ses lettest ved at betragte Ganttkortet i figur 2.

Antag nu, at aktivitet i og j opfylder (4), men at der nu defineres en ny rækkefølge σ' ved

$$\sigma'(i) = \sigma(j),\ \sigma'(j) = \sigma(i),\ \sigma'(k) = \sigma(k),\ k \neq i, j.$$

Lad $\{y'_1, \ldots, y'_m\}$ være summerne i (5) hørende til den nye rækkefølge. Klart nok vil vi have $y'_k = y_k$ for $k < \sigma'(j)$ og for $k > \sigma'(i)$. Antag, at der er et k imellem disse, så at $y'_k < y_k$. Det er let at se, at dette k må være det første eller andet mulige, dvs. $k = \sigma'(j)$ eller $k = \sigma`(j) + 1$, svarende til følgende to muligheder:

(1) Summen af $a_1^{\sigma'(h)}$ op til og med i er blevet mindre, dvs. $a_1^j < a_1^i$, således at $y'_{\sigma'(j)} < y_{\sigma(j)}$. Det følger af

$$a_2^j < \min\left\{a_1^j, a_2^i\right\},$$

at alle den anden sum i udtrykket for y_k er blevet endnu mindre, dvs. at y_k er vokset mere, end $y'_{\sigma'(j)}$ er faldet.

(2) Summen af $a_2^{\sigma'(h)}$ op til i er blevet større, svarende til at

$$\min\left\{a_1^i, a_2^j\right\}$$

antages i af a_1^i. Det ses på samme måde som ovenfor, at i dette tilfælde må $y'_{\sigma'(j)}$ være steget mere, end $y'_{\sigma'(j)+1}$ er faldet.

Vi konkluderer, at der må gælde $T_\sigma \leq T_{\sigma'}$. □

Vi kan nu nå den ønskede konklusion om vor algoritme:

Johnson's algoritme løser problemet om at finde korteste gennemløbstid i tilfældet m aktiviteter, 2 maskiner.

Vi har fra vort andet hjælperesultat ovenfor, at Johnson's algoritme ordner aktiviteterne således, at

$$\min\left\{a_1^i, a_2^j\right\} < \min\left\{a_1^j, a^j\right\}$$

netop når

$$\sigma(i) < \sigma(j).$$

Fra den tredie hjælpesætning har vi, at enhver ombytning af to aktiviteter fører til, at gennemløbstiden vokser (evt. er uforandret). Da enhver permutation kan fås frem ved gentagne parvise ombytninger, har vi det ønskede resultat. □

Johnson's algoritme lader sig ikke generelt overføre til mere komplicerede situationer, f.eks. til situationen med *tre* maskiner i stedet for to. Sådanne generelle flow-shops kræver andre – og typisk langt mere besværlige – metoder.

7. Opgaver

1. Firmaet Smashy Contractors bygger et kontorpalads i det tidligere Kongens Have. Byggeriet har følgende etaper:

1. Planlægning (4. mdr.)
2. Jordarbejder (4 mdr.)
3. Fundament og P-kælder (3 mdr.)
4. Tilkørselsfaciliteter (2 mdr.)
5. Parkeringspladser og beplantning (4 mdr.)
6. Montage af etager 1-10 (8 mdr.)
7. Montage af etager 11-38 (9 mdr.)
8. VVS og elinstallation (3 mdr.)

9. Marmorbeklædning af facader (6 mdr.)
10. Opstilling af moderne kunst på P-plads (2 mdr.)
11. Reklamekampagne (8 mdr.)

Af disse skal aktiviteterne i (1) være færdige før de øvrige kan påbegyndes. Tilsvarende skal (2) være slut før (3)-(10) kan starte, (3) skal gå forud for (6), (7), (8) og (9), (6) skal komme før (7), (8) og (9), og (7) skal komme før (8).

Marmorbeklædningen er finansieret af et EF-tilskud, betinget af, at der købes en bestemt type marmor fra Portugal. Beløbet (3 mill. ECU) udbetales fra EF til Smashy ved starten af byggeriet.

Smasky overvejer at anvende beløbet til valutaspekulationer indtil marmoret skal leveres. Pengene bindes derved i 3 måneder ad gangen, og den forventede indtjening er 36% for en sådan tre måneders periode. Når marmoret købes, betales det kontant.

Hvad kan Smashy tjene?

2. Konstruer netværket hørende til et projekt bestående af 7 jobs med følgende rækkefølgekrav:

A	før	B,C
B	før	D
E	før	A,D
A,B,D	før	F
D	før	G

Når varighederne af de enkelte jobs er:

A	B	C	D	E	F	G
17	23	34	22	19	17	16

ønskes projektets minimale varighed og den kritiske vej.

3. Ved en sygeeksamen er der tilmeldt otte eksaminander fra forskellige fag. Alle har krav på forberedelsestid, og der er derefter en eksamination af en bestemt længde. Såvel forberedelsestid som eksamination afhænger af faget, og i den konkrete situation er det som følger (i minutter):

Fag:	Forberedelse	Eksamination
1	18	10
2	20	20
3	10	15
4	13	10
5	10	12
6	25	30
7	10	10
8	20	40

Der er en fælles censor, som bliver betalt for antal minutter, som han er tilstede. Hvorledes skal eksamen tilrettelægges, for at denne betaling bliver mindst mulig?

4. Flyselskabet FnobusAir har koncession på at betjene et antal linier som vist i tabellen. De tre tal i tabellen angiver dels omkostningerne i 100.000 kr. pr. flyvning, dels koncessionens bestemmelser om mindste og største antal flyvninger pr. uge. Tabellen skal læses således, at oplysningerne i række i og søjle j refererer til flyvningen fra i til j (bemærk, at man har koncession på flyvning i en bestemt retning, men typisk ikke på den modsatte retning).

	A	B	C	D	F
A	$3; 2; 10$		$6; 4; 8$		
B				$8; 1; 9$	
C				$3; 3; 12$	$2; 3; 8$
D	$4; 2; 7$	$5; 3; 9$			
E		$4; 3; 8$		$2; 2; 8$	

Find den billigste trafikplan.

8. Litteratur

Emnet for dette kapitel, optimal tilrettelæggelse af produktionen ved store og komplekse projekter, dukkede op i 1950erne og 1960erne og blev blandt andet stimuleret af forsøg på at styre visse af de store ministerier i USA. En grundig behandling af scheduling i sine forskellige versioner kan findes i Conway, Maxwell og Miller (1967), og en bredere fremstilling med flere relationer til andre operationsanalytiske emner i Johnson (1974).

KAPITEL 13

Kompleksitet

1. Indledning

I de foregående kapitler har vi flere gange været inde på overvejelser om det vanskelige i at finde en løsning til et givet optimeringsproblem. Således har vi f.eks. i kapitel 2 diskuteret alternative lineære programmeringsalgoritmer, begrundet med at simplex-metoden trods sine kvaliteter i praksis havde visse teoretiske ømme punkter. Vi har også set eksempler på problemer, hvor det ville have være besværligt at bruge den generelle simplex-metode (max-flow, transport, assignment), fordi problemet havde en særlig struktur, som man med fordel kunne udnytte. Endelig har vi været inde på, at visse optimeringsmetoder (typisk indenfor heltalsoptimering) var "vanskelige", forstået sådan at der kunne gå meget tid med at finde en løsning ved hjælp af selv den smarteste algoritme.

I dette kapitel diskuterer vi disse ting lidt nærmere, hvilket (som man nok kunne ane) sker ved at vi formaliserer begrebet "vanskelig" i tilknytning til optimeringsmetoder; samtidig skifter vi så til det noget mere overbevisende navn "kompleksitet". Kapitlet afviger lidt fra de hidtidige, for der vil ikke blive snakket om nye metoder til konkret problemløsning; i stedet skal kapitlet ses som et lille supplement til den almene dannelse, specielt på det operationsanalytiske område. Sagen er den, at teorien om kompleksitet trods sin ikke særlig lange levealder – den fremkom omkring 1970 – allerede er så indarbejdet i faglitteraturen, at det i stadig stigende grad svirrer med udtalelser om, at denne eller hin metode er "polynomial" eller "NP-komplet". Hvad det så er for noget, får man at vide her; formålet med kapitlet er dels at forklare et (heldigvis overskueligt) antal fagudtryk, og at give så meget fornemmelse af, hvad det drejer sig om, at man vil være i stand til at forstå diskussionerne på feltet – så længe de ikke bliver alt for syrede.

2. Beregninger og deres kompleksitet

Som udgangspunkt er sværhed eller vanskelighed af en given opgave noget, man kender godt fra egen erfaring. På den anden side har man lige så megen erfaring for, at det, man selv synes var svært, forekom nemt for en anden stræber i ens

omgangskreds, så det virker ikke som om det er så nemt at måle. Det gør man egentlig heller ikke, så et eventuelt håb om ad objektiv vej at fastslå, at der var eksterne årsager til ens mangel på succes i den hidtidige akademiske karriere, må desværre skuffes. Man slår problemer sammen i ret store klasser, indenfor hvilke man ikke skelner; det afgørende er, hvilken af klasserne, som et givet problem skal puttes i.

Fra starten går man ret intuitivt frem (og bliver så mere og mere sofistikeret, efterhånden som man møder modstand). Hvis man skal måle, hvor svært det er at løse et problem, kan man prøve at lade en eller anden forsøgsperson løse det og måle tiden. Da man roder sig ind i en del subjektivt snavs ved at vælge en person, er der gode argumenter for at lade en robot løse problemet. Det skal selvfølgelig ikke være en hvilkensomhelst robot; en bilvaskerobot kan ganske vist vaske biler med og uden voksbehandling, undervognsvask og så videre, men den er ikke til megen hjælp hvis vi står med et Traveling Salesman problem snarere end med en snavset bil. Kort sagt, vi skal bruge en slags universalrobot, noget i retning af en PC, som kan fodres med vilkårligt mange data; det er dog ikke hensigtsmæssigt, at vor målemaskine er en konkret eksisterende maskine (f.eks. en IBM PC af årgang 1993), for den konkrete arkitektur af en sådan bliver meget hurtigt forældet. Den maskine, som vi bruger, er derfor en teoretisk computer, den såkaldte Turing-maskine, der på sin side er bedre end alle eksisterende maskiner (fordi den kan tage imod vilkårligt lange input-sekvenser), og på den anden side fremstår som noget primitiv, da den mangler alle de salgsfremmende dimser, som en moderne computer er udrustet med. Vi kommer tilbage til maskindiskussionen i løbet af et par afsnit.

Det næste skridt er så at finde et mål for, hvor lang tid det tager at løse på en computer. Her er der et par generende detaljer: Den tid, det tager, vil selvfølgelig afhænge af, hvilken metode man bruger til udregningen, dvs. af beregningsalgoritmen. Ved et banalt problem som addition af to tal er der jo ikke så mange algoritmer at vælge imellem, idet denne addition opfattes som en af de basale regneoperationer, som enhver computer klarer i ét trin (det sidste er strengt taget ikke helt rigtigt; addition er faktisk en hel række opdateringer af processor'ens registre, men det gøres på en standard måde i alle computere). Men f.eks. for et lineært programmeringsproblem er der, som vi så, flere forskellige algoritmer at vælge imellem, og det gør netop betydelig forskel, hvilken algoritme der vælges. Denne flertydighed vil vi klare ved at antage, at der overalt vælges den algoritme, som garanterer hurtigst beregning, således at den længste tid, der kan gå i noget specialtilfælde, er mindst mulig. Kompleksiteten af en problemtype er altså kompleksiteten ved den smarteste måde at løse problemet på.

Sidst, men ikke mindst, hvad er det vi måler, hvad forstås ved en problemtypes kompleksitet? Det kan næppe være et tal, for vanskeligheden ved at løse et problem afhænger klart af problemets størrelse. Derfor må det være en funktion af problemstørrelsen (antal parametre i en eller anden forstand, som vil blive præciseret hen ad vejen). Og som nævnt i tidligere skal vi ikke sondre mellem særlig mange klasser af funktioner. Den vigtigste sondring er mellem polynomier

og andre funktioner. Et polynomium $p(n)$ i n er et udtryk af formen

$$p(n) = a_k n^k + \cdots a_1 n + a_0$$

hvor a_i er reelle tal (typisk ikke-negative), for $i = 0, 1, \ldots, k$. Et polynomium i n vokser med n, endda ret hurtigt, hvis der er led af højere grad i n, men væksten er alligevel for intet at regne mod eksponentiel vækst, f.eks. funktionen 2^n. Hvis antallet af beregningstrin vokser eksponentielt, har vi at gøre med et stygt problem.

3. Decisionsproblemer

Når vi fordyber os lidt mere i teorien, vil vi i første omgang slet ikke betragte problemer som addition eller maximering, men problemer, hvis løsning er enten Ja eller Nej, såkaldte *decisionsproblemer*. Ofte er det ikke svært at transformere et givet problem til et decisionsproblem – det banale additionsproblem $m + n = ?$ kan jo laves om til: Givet m, n og k, er $m + n = k$? – og selvom problemerne ikke er identisk de samme vil de ofte vise sig at være ækvivalente i kompleksitetssammenhæng.

Lad os tage et andet eksempel, Traveling Salesman problemet, jvf. vor diskussion i kapitel 4: Givet en endelig mængde $M = \{1, \ldots, m\}$ af byer og for hvert ordnet par (i, j) af forskellige byer en omkostning c_{ij} ved at bevæge sig fra i til j, find en rute gennem alle byer som minimerer den samlede omkostning, dvs. en permutation π af mængden $M = \{1, \ldots, m\}$ så

$$\sum_{i=1}^{m} c_{(\pi(i),\pi(i+1))} + c_{(\pi(m),\pi(1))}$$

er mindst mulig. Dette er et som bekendt et heltalsprogrammeringsproblem, som ganske vist kan løses ved én for én at gennemgå alle permutationer π, noget der dog hurtigt forbyder sig selv, idet der jo er $m!$ af dem. Problem er ét af de ret besværlige, hvor der ikke findes gode algoritmer.

Foreløbig skal vi blot notere os, at vi kan få et decisionsproblem ud af Traveling Salesman ved for et givet tal B at spørge, om der findes en rute med afstand mindre end B. Med andre ord, vort decisionsproblem bliver: Givet mængde $M = \{1, \ldots, m\}$ og tallene c_{ij}, $i, j = 1, \ldots, m$, samt tallet B, findes der en rute (permutation π) således at

$$\sum_{i=1}^{m} c_{(\pi(i),\pi(i+1))} + c_{(\pi(m),\pi(1))} \leq B?$$

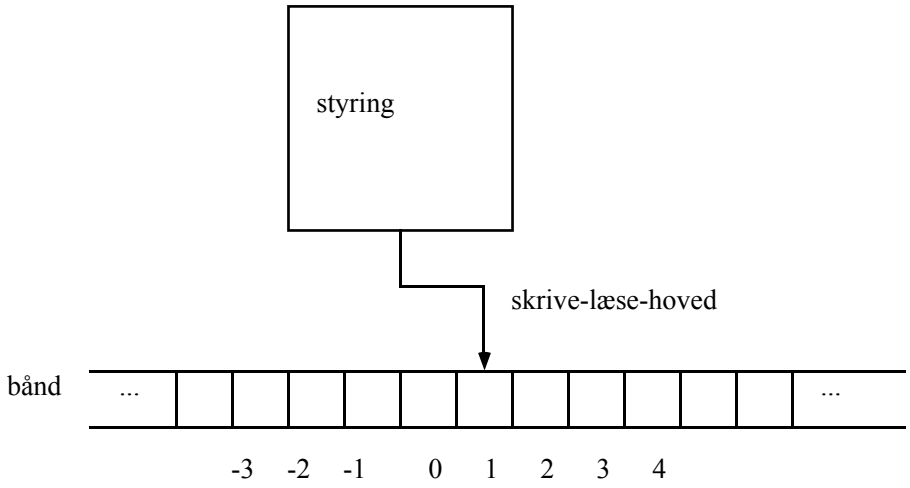
Tydeligt nok er det mindst lige så svært at løse minimeringsproblemet som at løse decisionsproblemet, så hvis det sidste er "svært" er optimeringsproblemet det også. Det kan iøvrigt vises, at de to problemer er i en vis forstand lige svære.

Idet vi stadig holder os til det intuitive niveau, kan vi notere os, at for to udgaver af Traveling Salesman decisionsproblemet ovenfor, hvor m^1 (antallet af byer i det første problem) er meget større end m^2 (antallet af byer i det andet problem), er det rimeligt at anse problem 1 som "større" end problem 2, ja faktisk kan vi bruge dette m som et mål på problemets omfang. Hvis vi nu forsøger at løse problemet ved den direkte metode, således at vi gennemgår samtlige ture og derefter svarer Ja eller Nej, skal vi udføre mindst $m!K$ operationer, hvor K er et mål for de operationer, som er involveret i addition af størrelserne $c_{(\pi(i),\pi(i+1))}$ og sammenligningen med B (dette K vil typisk afhænge lineært af m), og for selv ret små m bliver det et astronomisk tal. Da $m!$ er af samme størrelsesorden som 2^m ser vi, at kompleksiteten af vor beregningsmetode er eksponentiel, hvilket er et udtryk for, at det for store m tager for lang tid at løse.

I praksis vil enhver løsning af et decisionsproblem via en computer kræve en kodning af problemet til symboler som computeren kan læse. Dette kan tænkes at ske på mange forskellige måder, og det betyder faktisk ikke så meget, præcis hvilken kodning der benyttes. Vi skal nemlig ikke interessere os for et mål i form af et bestemt tal som udtryk for kompleksiteten, men for om beregningstiden er en ikke særlig drastisk voksende funktion af problemets størrelse, eller den omvendt vokser meget hurtigt, således som det kan vises at være tilfældet for Travelling Salesman problemet. Som følge heraf er det heller ikke så meget de enkelte operationer i algoritmerne, der interesserer os, som eksistensen af en eller anden beregningsmetode (hvor mærkelig den så måtte være) der klarer problemet som en pæn funktion af dets størrelse.

Selvom den specifikke kodning ikke interesserer os så meget, må vi opholde os lidt ved den grundlæggende struktur. Det oprindelige problem oversættes ved kodningen til en lang liste af symboler (f.eks. en liste af 0 og 1 og mellemrum). Disse fodres så ind i en maskine (eller maskinen tager imod andre symboler, f.eks. ASCII, men oversætter så selv til 0 og 1). Maskinen går da igang, og efter at have behandlet vort kodede decisionsproblem ender den med at sige "Ja" eller "Nej" (med de symboler, der er til rådighed). Altså: vort oprindelige problem er kogt ned til at der er visse ganske bestemte symbollister, som maskinen skal reagere med et "Ja" på indenfor en vis tid (som er udtryk for kompleksiteten), mens alle andre lister får et "Nej". I faglitteraturen bruges en bestemt terminologi for disse ting: Vi har den givne mængde af symboler, der kaldes for et *alfabet* (i vort eksempel bestående af 0 og 1). En liste af symboler fra alfabetet kaldes et *ord* (over det givne alfabet), og en mængde af ord kaldes et *sprog*. Hvis en bestemt maskinen ender med "Ja" netop når den fodres med ordene i et givet sprog, siger vi, at den accepterer dette sprog. Man kan således opfatte maskinens rolle som at den skal genkende en bestemt struktur i et ord, svare "Ja" hvis det hører til sproget, "Nej" ellers.

Som man kan fornemme, er denne terminologi brugbar ved studiet af maskinel bearbejdning af (sædvanlige) sprog eller af billeder. Her skal vi koncentrere os om en bestemt slags maskiner og sprog accepteret af disse.



Figur 1

4. Turing maskiner

Til løsning af generelle decisionsproblemer (eller snarere, til "genkendelse" af bestemte sprog) vil vi benytte en slags idealiseret computer, som iøvrigt er opfundet før de praktisk anvendelige, nemlig af A.Turing i 1936. På en simpel måde sammenfatter denne, hvad en computer kan, og som sætter sig ud over den konkrete computers kapacitetsbegrænsninger. Essentielt er Turing maskinen en slags automat, som reagerer på et input ved at producerer et ganske bestemt output. Den afgørende finesse i Turing maskinen, det der gør den til en computer, er, at den er i stand til at modificere sine instrukser.

Vi kan forestille os en Turing maskine som bestående af

(1) styringsdel med endelig mange tilstande
(2) et skrive-læse-hoved
(3) et bånd af uendelig længde, bestående af celler nummeret

$$\ldots -3, -2, -1, 0, 1, 2, 3, \ldots ;$$

båndet kan bevæges i begge retninger:
Et *program* for Turing maskinen består af følgende:

(a) En endelig mængde af *båndsymboler*, dvs. et alfabet I, som skal indeholde en delmængde Σ af *input symboler* og et specielt mellemrumssymbol $\nu \in I \backslash \Sigma$.
(b) En endelig mængde af tilstande Q, indeholdende en specificeret *start-tilstand* q_o og to stop-tilstande q_J og q_N.
(3) *En overgangsfunktion* $\delta : (Q \backslash \{q_J, q_N\}) \times I \to Q \times I \times \{-1, +1\}$.

Vi bruger lidt (standard) notation og terminilogi: Hvis A er en mængde af symboler, kan vi betragte alle mulige endelige følger af symboler fra A. Denne mængde skrives A^*, og dens elementer kaldes *ord*. Analogien med sædvanlige alfabeter og ord er oplagt, men i vor sammenhæng er alle bogstavkombinationer ord.

Programmet fungerer som følger: Maskinen starter i q_o med skrivelæsehovedet ved celle 1. Hvis maskinens tilstand er enten q_J eller q_N, stopper beregningen, og det endelige svar er henholdsvis "Ja" og "Nej". I modsat fald er maskinens tilstand q i $Q\backslash\{q_J, q_N\}$, og overgangsfunktionen δ specificerer da til denne tilstand q og det symbol, der står i celle 0 (som eventuelt kan være ν) en ny tilstand q' for styringsdelen, et nyt symbol s' som skrives i stedet for det s, der stod i cellen, og en bevægelse af skrive-læse-hovedet, en celle til venstre hvis der står -1 på tredie plads i $\delta(q, s)$, en celle til højre hvis der står $+1$.

Beregningen er afsluttet, hvis maskinen på et eller andet tidspunkt kommer i tilstanden q_J i hvilket tilfælde vi siger, at maskinen har *accepteret* det ord $w \in \Sigma^*$, der stod på båndet ved starten, eller hvis maskinen kommer i tilstanden q_N, hvorved ordet w er *forkastet*. Men det er ikke udelukket, at maskinen slet ikke stopper.

Vi ser, at der er en tæt forbindelse mellem maskinens umiddelbare funktion, som er at acceptere eller forkaste ord, dvs. at genkende sprog ligesom de endelige automater i forrige kapitel, og decisionsproblemerne, idet disse sidste efter kodning fremstår som ord, der accepteres af maksinen, hvis svaret er "Ja".

Maskinen kan faktisk bruges til mere end dette, nemlig til at beregne funktioner. Hvis vi har en maskine, der stopper for alle ord w i en vis delmængde af Σ^*, får vi derved en funktion f fra alle sådanne ord w til ord $f(w)$ i I^*, idet vi som funktionens værdi tager alle de symboler, som står i celle $1, 2, 3 \ldots$ osv. frem til første celle med et ν.

Trods den tilsyneladende noget besværlige funktion af Turing maskinen er den faktisk i stand til at udføre alt hvad eksisterende computere kan (når programmet vælges på passende måde), og den kan således opfattes som en idealisering af både computere og den menneskelige hjerne, det sidste i det mindste hvad angår beregningsarbejde.

Vi kan nu definere (tids-)kompleksitet af beregning ved at tælle antallet af skridt inden maskinen stopper ved q_J eller q_N: For et program M i en Turing maskine, konstrueret således at det stopper for alle inputs, er *tidskompleksitetsfunktionen* $T_M : N_o \to N_o$ givet ved

$$\begin{aligned} T_M(n) = \max\{m| \text{ der findes } x \in \Sigma^*, \text{ med } \ell(x) = n, \text{ så} \\ \text{beregningen stopper i } q_J \text{ eller } q_N \text{ efter } m \text{ skridt } \} \end{aligned}$$

Et program M i en Turing maskine siges at være et *polynomialt tids program* hvis der findes et polynomium p således at $T_M(n) \leq p(n)$ for alle n.

Fra disse overvejelser, der alle har at gøre med at acceptere sprog, kan vi gå videre og isolere en klasse af særlig "pæne" problemer: Lad L være et sprog, dvs.

Eksempel 13.1

Nedenfor er angivet et Turing maskine program, der accepterer alle ord, som (læst fra venstre mod højre) ender med to nuller:

$$\Gamma = \{0, 1, b\},\ \Sigma = \{0, 1\}$$

$$Q = \{q_0, q_1, q_2, q_3, q_J, q_N\}$$

Tabellen nedenfor angiver overgangsfuntionen δ:

	0	1	b
q_0	$(q_0, 0, +1)$	$(q_0, 1, +1)$	$(q_1, b, -1)$
q_1	$(q_2, b, -1)$	$(q_3, b, -1)$	$(q_N, b, -1)$
q_2	$(q_Y, b, -1)$	$(q_N, b, -1)$	$(q_N, b, -1)$
q_3	$(q_N, b, -1)$	$(q_N, b, -1)$	$(q_N, b, -1)$

En beregning med programmet kan se ud som følger (hvor vi har givet input 10100: Vi starter i tilstand q_0 med båndet

$$\text{Tilstand } q_0 \ \cdots \mid b \mid 1 \mid 0 \mid 1 \mid 0 \mid 0 \mid b \mid \ \cdots \text{ læser celle 2}$$

hvor der læses fra den første ikke-blanke celle fra venstre, altså fra den viste celle 2 (fra venstre). Der læses nu 1 og vi går derfor ifølge tabellen til næste celle mod højre (celle 3), idet vi bliver i tilstand q_0 og lader 1-tallet stå:

$$\text{Tilstand } q_0 \ \cdots \mid b \mid 1 \mid 0 \mid 1 \mid 0 \mid 0 \mid b \mid \ \cdots \text{ læser celle 3;}$$

igen flyttes en plads til højre, så vi får

$$\text{Tilstand } q_0 \ \cdots \mid b \mid 1 \mid 0 \mid 1 \mid 0 \mid 0 \mid b \mid \ \cdots \text{ læser celle 4,}$$

og så fremdeles gennem følgende trin:

$$\text{Tilstand } q_0 \ \cdots \mid b \mid 1 \mid 0 \mid 1 \mid 0 \mid 0 \mid b \mid \ \cdots \text{ læser celle 5,}$$

$$\text{Tilstand } q_0 \ \cdots \mid b \mid 1 \mid 0 \mid 1 \mid 0 \mid 0 \mid b \mid \ \cdots \text{ læser celle 6,}$$

$$\text{Tilstand } q_0 \ \cdots \mid b \mid 1 \mid 0 \mid 1 \mid 0 \mid 0 \mid b \mid \ \cdots \text{ læser celle 7.}$$

Nu læser maskinen en blank celle, og det får den til at skifte tilstand til q_1 og gå en celle tilbage:

$$\text{Tilstand } q_1 \ \cdots \mid b \mid 1 \mid 0 \mid 1 \mid 0 \mid 0 \mid b \mid \ \cdots \text{ læser celle 6;}$$

nullet i den læste celle viskes ud, der skiftes tilstand til q_2 og gås endnu et skridt tilbage:

$$\text{Tilstand } q_2 \ \cdots \mid b \mid 1 \mid 0 \mid 1 \mid 0 \mid b \mid b \mid \ \cdots \text{ læser celle 5,}$$

hvor der læses endnu et nul; det får maskinen til at gå i q_J, samtidig med at den iøvrigt visker nullet ud og rykker endnu en celle tilbage:

$$\text{Tilstand } q_J \ \cdots \mid b \mid 1 \mid 0 \mid 1 \mid b \mid b \mid b \mid \ \cdots \text{ læser celle 4.}$$

Eksempel 13.1, fortsat

Det afgørende er, at maskinen har accepteret vort input, idet den jo er endt i tilstand q_J. Vi kan iøvrigt af denne beregning se, hvordan programmet virker: Først kører vi inputtet igennem, indtil vi er sikker på at have scannet sidste symbol. Så checker vi om dette sidste symbol er et 0; hvis det er det, skifter vi tilstand ("sidste symbol 0 verificeret") og gentager proceduren, idet vi visker dette symbol ud og rykker tilbage. Hvis også det nye sidste symbol er et 0, er sagen i orden.

Eksemplets program var naturligvis overordentlig simpelt; generelt er programmer for Turing-maskiner en del vanskeligere at gennemskue umiddelbart, og det er egentlig heller ikke formålet. Pointen er som regel, at der *findes* et program, der gør et eller andet, i dette tilfælde checker for to nuller i enden. Turing maskiner kan langt mere, hvilket berettiger til deres centrale placering i denne type overvejelser.

en delmængde af Σ^*, som er fremkommet ved indkodning af alle tænkelige tilfælde af et givet beregningsproblem. Hvis M er et program i en Turing maskine, som stopper for alle input i Σ^* og som stopper i q_J netop for de ord w, som er i L, således at mængden L_M af de ord, som accepteres af maskinen, er netop L, siger vi, at programmet M genkender L; vi definerer da mængden

$$P = \{L| \text{ der findes et polynomialt tids program } M \text{ med } L_M = L\}.$$

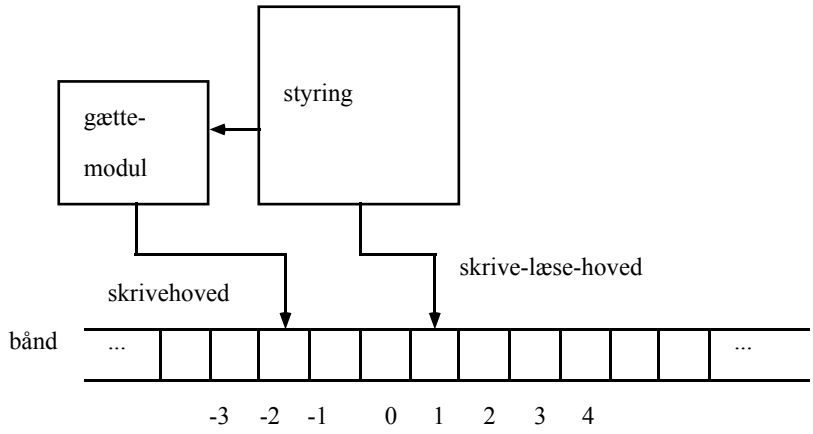
Vi siger, at problem tilhører P hvis det under en eller anden kodning går over i et sprog i P.

Problemer i P, også kaldt *polynomiale problemer* er at opfatte som særdeles simple – eller omvendt, problemer, som ikke er i P, er sådanne, hvis løsning for store n kan tage overordentlig lang tid. Ved en nøjere gennemgang af enkelte problemer viser det sig, at man for mange "gængse" problemer – som f.eks. vor ven Travelling Salesman" *ikke* kan vise, at de tilhører P, idet man indtil videre ikke har kunnet angive et polynomialt tidsprogram. Meget tyder på, at det heller ikke vil være muligt. Vi skal derfor gå et skridt videre og betragte en klasse af problemer, der med hensyn til kompleksitet ligger et trin over de polynomiale.

5. Ikke-deterministiske Turing maskiner og klassen NP

For at motivere interessen for en anden og mere generel type af Turing maskiner vil vi et øjeblik vende tilbage til Traveling Salesman. Som nævnt i forrige afsnit har det hidtil ikke været muligt at finde en algoritme, der i polynomial tid kan afgøre alle Traveling Salesman decisions problemer. På den anden side, hvis vi har et givet problem, specificeret ved byer, afstande, og en vis begrænsning B, og vi får opgivet et forslag til en rute, er det en forholdsvis simpel sag at *verificere*, at forslaget faktisk er en rute, og at denne rute opfylder begrænsningen. Denne verifikationsprocedure kan faktisk gennemføres i polynomial tid.

Det er klart, at verifikation af et forslag til løsning ikke er det samme som en systematisk procedure til at finde løsningen; man springer et led over, som overlades



Figur 2

til et gæt. Når man udregner tiden brugt på verifikation af en (gættet) løsning og ikke medtager tiden brugt på at søge en sådan, har vi selvfølgelig udeladt et ikke uvæsentligt af, hvad der indgår i en realistisk løsningsprocedure. Det viser sig dog alligevel, at der kommer noget frugtbart ud af denne betragtningsmåde.

Teknisk definerer vi klassen af lidt-sværere-end-polynomiale problemer ved hjælp af *ikke-deterministiske Turing maskiner*, der i det store og hele er opbygget som de almindelige ("deterministiske") Turing maskiner fra forrige afsnit, bortset fra at de er udstyret med et "gætte-modul", som klarer den indledende fase. Vi kan anskue en ikke-deterministisk Turing maskine som følger:

Den tekniske specifikation af et program er den samme som før, idet den stadig består af bånd-alfabet I, input alfabet Σ, mellemrumssymbol ν, tilstande Q med starttilstand q_o og sluttilstande q_J og q_N, samt overgangsfunktionen

$$\delta : (Q \backslash \{q_J, q_N\}) \times I \rightarrow Q \times I \times \{-1, +1\}.$$

Det er egentlig kun i fortolkningen af, hvad der foregår under en beregning, at der er forskel.

Beregningen på en ikke-deterministisk Turing maskine foregår i to trin. Første trin er et "gæt". I starten er input ordet w skrevet i cellerne fra 1 op til (w) og alle andre celler er tomme. Skrive-læse-hovedet hørerende til styringsdelen står ved celle 1, gættemodulets skrivehoved ved celle -1, og styredelen er inaktiv. Gættemodulet beordrer så skrivehovedet til at skrive et symbol fra I i cellen og rykke en celle til venstre, eller til at stoppe. Ved den sidste ordre stopper gætningen, styringsdelen går i gang, og beregningen fortsætter ligesom for en deterministisk Turing maskine. Denne beregning stopper, hvis maskinen kommer i tilstand q_J eller q_N; i første tilfælde har maskinen accepteret ordet w. Bemærk, at for et givet input w er der uendelig mange beregningsforløb, nemlig et for hvert muligt ord tilføjet i gættefasen. Hvad der kræves for at ordet w anses for accepteret er at ét af disse forløb ender i q_J.

Fra nu af kan begreberne defineres helt som i forrige afsnit: Sproget L_M defineres som alle de ord i Σ^*, der accepteres af M; Tiden anvendt til beregningen sættes til at være antallet af skridt brugt i *såvel* gætte- som beregningsfasen. Endelig har vi tidskompleksitetsfunktionen $T_M : N_o \to N_o$ givet ved

$$T_M(n) = \max\{\{1\} \cup \{m| \text{ der findes } x \in L_M \text{ med } \ell(x) = n \text{ så at tiden brugt til accept af } x \text{ er } m.\}\}$$

Bemærk, at der kun regnes med tiden brugt i input, som ender med accept af x; hvis der slet ikke er sådanne ord af længde n, sættes $T_M(n)$ pr. konvention til 1. At man kun ser på input, som accepteres, hænger sammen med, at accept og forkastelse ikke optræder helt symmetrisk i ikke-deterministisk beregning (det gjorde de i forrige afsnit). Vi skal ikke gå nærmere ind på disse detaljer.

Vi kan nu definere et polynomial tids-program for en ikke-deterministisk Turing maskine ved at $T_M(n) \leq p(n)$, alle $n > 1$, for et eller andet polynomium, p, og vi har da en klasse

$$NP = \{L| \text{ der findes et polynomialt program i en ikke-deterministisk Turing maskine, som genkender } L\}$$

af sprog, som kan genkendes i polynomial tid af en ikke-deterministisk Turing maskine. Vi identificerer som sædvanlig problemer med sprog gennem passende kodning og kan således tale om, at det og det problem hører til klassen NP.

6. Polynomiale transformationer og NP-komplette problemer

Vi har på nuværende tidspunkt etableret et hierarki af problemer, således at vi nederst har de polynomiale, der kan anses for rimeligt håndterlige. Ovenover disse har vi så klassen NP, som er en større klasse; hvis et problem tilhører P og altså accepteres af en deterministisk Turing maskine, vil den så meget mere kunne accepteres af en ikke-deterministisk maskine og er altså i NP. Men der er problemer, som er i NP og ikke i P: Vort hårdt prøvede eksempel Traveling Salesman er et sådant.

Det næste spørgsmål, som melder sig naturligt, er om denne inddeling i "sværhed" eventuelt kan gøres finere, således at man f.eks. vil kunne sondre imellem problemer indenfor $NP \backslash P$. Som et første trin til at sondre imellem problemer har vi behov for et mål for, at to forskellige problemer er lige svære; her er det intuitivt oplagt at benytte som kriterium, at det ene problem "let" skal kunne transformeres til det andet, og omvendt. Hvis vi hertil føjer, at vi ved "let " vil forstå, at denne transformation skal kunne foretages af en deterministisk Turing maskine i polynomial tid, har vi et rimeligt kriterium for at sammenligne sværhed af problemer. Bemærk, at vi allerede i afsnit 1.2 noterede os Turing maskinens mulighed for at beregne funktioner.

Vi siger følgelig, at sprog L_1 kan *transformeres polynomialt* til et sprog L_2, skrevet $L_1 \propto L_2$ hvis $L_1 \subset \Sigma_1^*$, $L_2 \subset \Sigma_2^*$, og der findes en funktion $f : \Sigma_1^* \rightarrow \Sigma_2^*$ som opfylder

(a) der eksisterer et polynomial tids program som beregner f
(b) $x \in L_1$ hvis og kun hvis $f(x) \in L_2$.

To sprog er ækvivalente hvis $L_1 \propto L_2$ og $L_2 \propto L_1$.

Begrebet polynomiale transformationer kan nu benyttes til at isolere en klasse af "særlig svære" NP-problemer. Vi siger, at et sprog L er *NP-komplet* hvis det er NP, og der for alle sprog L' i NP gælder, at $L' \propto L$. Endelig identificerer vi som vanligt problemer med sprog.

At et problem er NP-komplet, betyder, at det med meget stor sandsynlighed ikke er polynomialt. For hvis det skulle vise sig muligt at finde en algoritme som løste problemet i polynomial tid, da ville man samtidig have fundet en polynomial algoritme (nemlig transformation + den nye algoritme) for *ethvert* problem i NP, hvilket ville være noget i retning af en historisk begivenhed. Bemærk dog, at dette sidste faktisk ikke er udelukket. Man har ikke noget bevis for, at bare et eller andet NP problem ikke har nogen polynomial algoritme; teorien er så at sige ikke færdig endnu.

At vise for et konkret problem, at det tilhører klassen af NP-komplette problemer, betyder, at man dels må angive et polynomialt program for en ikke-deterministisk Turing maskine, som løser dette problem (herved har man sikret sig, at problemet ikke er sværere end NP) og dernæst, at *alle* NP problemer kan transformeres til det givne. Dette kan være no så vanskeligt, men man er i praksis hjulpet ved, at det er nok at vise, at ét NP-komplet problem kan transformeres; relationen $\propto$ er transitiv.

7. SATISFIABILITY og Cook's sætning*

Indtil nu har vi ganske vist indført et omfattende begrebsapparat, men vi har ikke vist noget og er derfor principielt ikke blevet meget klogere. Der mangler en viden om, at et eller andet konkret decisionsproblem, enten Traveling Salesman eller et andet, er NP-komplet. Har vi først det, kan vi få en hel række andre ved at vise, at dette konkrete problem kan transformeres polynomialt til andre. Nøglen til den videre succes er derfor et resultat om bare et enkelt decisionsproblem.

Denne nøgle blev leveret af Cook i 1971, og resultatet, der som det allerede er fremgået er helt centralt for kompleksitetsteorien, hedder derfor Cook's sætning.

Vi vil i det følgende give en noget populariseret version af Cook's resultat, hvor der skøjtes hen over en del besværlige detaljer. Det kan imidlertid stadig give en vis fornemmelse af, hvordan kompleksitetsteorien er skruet sammen.

Det konkrete problem, som behandles, virker umiddelbart ikke som et, man ville være faldet over i dagligdags problemer. Det skal man dog ikke lade sig narre af, for som angivet ovenfor er det afgørende ikke, om problemet er hverdagsagtigt, men om det let lader sig omforme til hverdagsagtige problemer, og det gør det faktisk.

Problemet, der går under betegnelsen SATISFIABILITY (der er tradition i denne teori for at skrive problemnavne med store bogstaver) eller kortere, SAT, kræver lidt ny terminologi: Vi starter med et antal variable $U = \{u_1, \ldots, u_m\}$, der tager værdier 0 eller 1 (det kalder man ofte for Boole'ske variable). Vi vil foruden hver af disse variable u_i også operere med deres negation $\bar{u}_i$ fortolket som "ikke-u_i". En funktion t, der til hver variabel giver enten T (sand) eller F (falsk), kaldes for en tildeling af sandhedsværdi (truth assignment).

En formel (eng.: clause) over U er et udtryk af typen $\{u_2, \bar{u}_5, u_6\}$ fortolket som " u_2 eller ikke-u_5 eller u_6", dvs. formlen tilfredsstilles af enhver tildeling t, som ikke har $t(u_2) = F, t(u_5) = T, t(u_6) = F$. Vi kan nu formulere SAT: Givet en mængde C af formler over U, findes der en tildeling t, som tilfredsstiller alle formlerne i C?

Der gælder nu følgende resultat (Cook, 1971), der kan ses som grundlæggende for hele kompleksitetsteorien:

SATISFIABILITY *er* NP-*komplet*.

Et stærkt skitseret bevis for Cook's sætning går som følger: Det er let at se, at SAT er i klassen NP. Man skal nemlig her blot vise, at hvis der er gættet på en tildeling t, da er der et polynomialt Turing maskine program, der checker om t tilfredsstiller alle formler i C. Hele vanskeligheden ligger i at vise, at der for alle NP-sprog L gælder $L \propto L_{SAT}$. Det går vi så i gang med.

Lad os derfor starte med et vilkårligt sprog L, som er NP, hvilket vil sige, at der er et ikke-deterministisk Turing maskine program, som genkender det. Vi kalder programmet for M, og det består som tidligere af Γ, Δ, b, Q, herunder $q_0, q_{J,N}$, og δ. Vi ved, så hvis input $x \in \Sigma^*$ accepteres af M, er der en beregning, hvor antallet af skridt i såvel gætte-del som regnedel ikke overstiger $p(n)$, hvor p er et polynomium og $n = |x|$, længden af x. Specielt kan maskinen aldrig have brugt andre celler end dem nummereret fra $-p(n)$ til $p(n) + 1$.

Det, vi nu skal gøre, er at angive en transformation af sproget til SAT (strengt taget skulde vi transformere til en mængde af ord som er kodede udgaver af SAT, men det springer vi over, da den specifikke kodning som altid er uvæsentlig). Til det formål indfører vi en hel stak af variable, som tilsammen skal være mængden U. Det drejer sig om følgende:

variabel	definitionsområde	fortolkning
$Q[i,k]$	$0 \leq i \leq p(n)$ $0 \leq k \leq r$	På tidspunkt i er M i tilstand q_k
$H[i,j]$	$0 \leq i \leq p(n)$ $-p(n) \leq j \leq p(n)+1$	På tidspunkt i er l.-s.-hovedet ved at læse celle j
$S[i,j,k]$	$0 \leq i \leq p(n)$ $-p(n) \leq j \leq p(n)+1$ $0 \leq k \leq \nu$	På tidspunkt i er indholdet i celle j symbolet s_k

En beregning ved M svarer faktisk til en passende tildeling af sandhedsværdi til hver af disse variable. Man ser nu, hvor det bærer henad, og det bliver endnu klarere, når vi derefter noterer os, at det naturligvis ikke er alle tildelinger, der modsvarer en beregning ved M. Hvad vi derfor skal gøre for at lave f_L, der omformer L til en udgave af SAT, er at opstille en familie af formler, der alle skal være tilfredsstillet af en tildeling for at denne svarer til en beregning med M.

Opstillingen af denne familie af formler er en lidt omstændelig affære, og her vil vi tillade os at snyde ved kun at angive starten (en fuld redegørelse kan findes enten i Cook's originale papir eller i den klassiske fremstilling, som er Garey og Johnson (1979)). Formlerne kan opdeles i seks grupper $G_1, \ldots, G_6$. Den første gruppe G_1 indeholder formlerne

$$\{Q[i,0], Q[i,1], \ldots, Q[i,r]\},\ 0 \leq i \leq p(n),$$

$$\{Q[\bar{i},j], Q[\bar{i},j']\},\ 0 \leq i \leq p(n),\ 0 \leq j \leq j' \leq r.$$

Disse formler siger blot det simple, at M dels er i mindst en eller anden tilstand, dels ikke i mere end én tilstand ad gangen. Ikke chokerende i sig selv, men betingelser, der skal med, hvis vi skal beskrive M helt ved formler og tildeling af sandhedsværdi. Helt tilsvarende skal vi i gruppen G_2 skrive betingelser for, at læse-skrive-hovedet på hvert tidspunkt i ser på netop en celle (her bruges H-variablene), og i gruppe G_3 at hver celle indeholder netop ét symbol fra Γ (hvortil vi bruger S-variable). I gruppe G_4 skal vi udtrykke, at beregningen på tidspunkt 0 er i den initiale tilstand for checke-delen af programmet, og G_5, der indeholder den ene formel $\{Q[p(n),1]\}$, fortæller, at på tidspunkt $p(n)$ er M i tilstand q_Y (og har accepteret x). Endelig er der G_6, som skal udtrykke, at på hvert tidspunkt i følger konfigurationen på tidspunkt $i+1$ ved én anvendelse af overgangsfunktionen δ fra konfigurationen på tidspunkt i. Også dette kan udtrykkes ved variablene af type H, Q og S og deres negationer.

Dermed har vi (bortset fra en del detaljer) angivet en transformation fra L til SAT. Det eneste der mangler, er at vise at denne transformation kan gennemføres i polynomial tid. Det er imidlertid en simpel konsekvens af vor fremgangsmåde, for vi har netop angivet alle de formler, der skal skrives op, og når man har n og $p(n)$,

kan man udregne, hvor mange variable og hvor mange formler, der kan blive tale om, og det kan ikke løbe op til mere end omkring $p(n)^4$, som selv er et polynomium. Dermed er vi ved det ønskede resultat. (□)

8. Andre NP-komplette problemer

Hvis vor historie stoppede ved Cook's sætning og NP-kompletheden af SAT, havde vi ganske vist fundet ud af, at der fandtes NP-komplette problemer, men klassen af sådanne kunne stadig tænkes at være temmelig eksotisk. Det er imidlertid ikke tilfældet. Når man først ved, at SAT er NP-komplet, kan man gå videre til en række andre problemer. Teknikken er næsten altid den samme: For et givet problem overbeviser man sig først om, at det tilhører klassen NP, og det er som regel ligetil. Dernæst viser man, at et kendt NP-problem kan transformeres polynomialt til det givne, og man er færdig.

3SAT. Det er praktisk som det første nye problem at tage et, der er ret meget lig det, vi lige har haft: Problemet 3SAT er givet ved en mængde C af formler over variablene i U, således at hver formel kun indeholder 3 variable (eller deres negation), og det gælder ligesom tidligere om at finde en tildeling af sandhedsværdier, der tilfredsstiller samtlige formler.

Opgaven er ifølge det foregående at vise, at SAT kan transformeres til 3SAT. Da vi i SAT har formler med mere end 3 variable, skal vi finde nogle nye variable, som vi kan bruge til at omformulere de givne formler til 3-formler. Lad U og $C = \{c_1, \ldots, c_m\}$ være henholdsvis variable og formler i en udgave af SAT. For et givet j skriver vi

$$c_j = \{z_1, \ldots, z_k\},$$

hvor z_i'erne er variable eller deres negationer. Der er flere tilfælde afhængig af k's størrelse, og i hvert tilfælde supplerer vi med en mængde U_j af nye variable og laver en mængde C'_j af 3-formler:

(1) $k = 1$. Vi sætter $U'_j = \{y_j^1, y_j^2\}$, hvor y_j^1 og y_j^2 er nye variable, som vi har opfundet på stedet, og vi indfører de fire nye formler i C'_j som

$$\{z_1, y_j^1, y_j^2\}, \{z^1, y_j^1, \bar{y}_j^2\}, \{z_1, \bar{y}_j^1, y_j^2\}, \{z_1, \bar{y}_j^1, \bar{y}_j^2\}$$

(2) $k = 2$. Vi sætter $U'_j = \{y_j^1\}$ og lader C'_j bestå af $\{z_1, z_2, y_j^1\}$ samt $\{z_1, z_2, \bar{y}_j^1\}$.

(3) $k = 3$. Her behøver vi ikke nye variable, og vi lader C'_j bestå af formlen c_j.

(4) $k > 3$. Mængden af nye variable er $\{y_j^i | 1 \leq i \leq k-3\}$, og de nye formler er dels $\{z_1, z_2, y_j^1\}$, dels alle formler $\{\bar{y}_j^i, z_{i+2}, y_j^{i+1}\}$ for $1 \leq i \leq k-4$, og endelig $\{\bar{y}_j^{k-3}, z_{k-1}, z_k\}$.

Der resterer nu at checke, at den samlede mængde af 3-formler over den nye udvidede mængde af variable er kan tilfredsstilles af en tildeling af sandhedsværdier, netop når den oprindelige mængde af formler kunne. Det vil vi ikke gå i detaljer med; for nogle af tilfældenes vedkommende er det ret oplagt, idet man tilføjer variable på en sådan måde, at man netop er dækket ind; for andre er det måske ikke helt indlysende, men det kan let checkes. Endelig skal transformationen være polynomial, hvilket er ret ligetil, eftersom vi har angivet en forskrift for, hvorledes man laver de nye formler, og antallet af nødvendige nye variable og formler er polynomialt begrænset. Alt ialt har vi derfor, at 3SAT er NP-komplet.

Overdækningsproblemet VC. Det næste problem, vi vil se på, fører os tilbage til noget, vi har set tidligere, nemlig en overdækning i en graf (V, E), dvs. en delmængde V' af V således at hver kant i E er incident med mindst et punkt i V'. Problemet er givet ved grafen (V, E) og et tal K, der naturligvis skal være $\leq |V|$, antallet af punkter i grafen, og spørgsmålet er så, om der findes en overdækning V' med ikke over K punkter. Dette problem betegnes VERTEX COVER eller blot VC. Vi viser (i hovedtræk), at VC er NP-komplet.

Igen er fremgangsmåden at transformere et kendt NP-problem til VC. Nu har vi også 3SAT til vor rådighed, så den bruger vi. Vi starter derfor med en mængde $C = \{c_1, \ldots, c_m\}$ af 3-formler over en mængde U af variable, og skal konstruere en graf, der har en overdækning mindre end K netop hvis C kan tilfredsstilles.

For hver variabel $u_i \in U$ indfører vi et lille stykke graf, nemlig delgrafen $T_i = (V_i, E_i)$ bestående af to punkter $u_i, \bar{u}_i$ og en kant mellem u_i og $\bar{u}_i$. Enhver overdækning må dermed indeholde enten u_i eller $\bar{u}_i$, for ellers er kanten i E_i ikke med.

For hver formel $c_j \in C$ indfører vi en komponent $S_j = (V'_j, E'_j)$ bestående af tre ny dannede punkter

$$V'_j = \{a_1[j], a_2[j], a_3[j]\}$$

alle de kanter, der forbinder dem, skrevet som

$$E'_j = \{(a_1[j], a_2[j]), (a_1[j], a_3[j]), (a_2[j], a_3[j])\}.$$

Enhver overdækning må indeholde mindst to punkter fra V'_j for at dække kanterne fra E'_j.

Indtil nu var konstruktionen helt mekanisk, og ganske uafhængig af, hvordan formlerne i C faktisk så ud. Denne information bruges til at forbinde komponenterne. For hver formel c_j skrives dens variable (evt. negationer) som x_j, y_j, z_j. Vi lader nu følgende kanter udgå fra komponenten S_j:

$$(a_1[j], x_j), (a_2[j], y_j), (a_3[j], z_j).$$

Vi sætter endelig K til $n + 2m$ og lader grafen (V, E) være den bestående af alle de konstruerede komponenter med de sidste kanter tilføjet.

Der resterer nu kun de sædvanlige to checkpunkter: Vi skal vise, at den konstruerede udgave af VC faktisk kan løses netop når 3SAT-problemet er løsbart, og vi skal sikre os, at konstruktionen kan gennemføres i polynomial tid. Begge disse overvejelser vil vi springe over her.

Andre NP-komplette problemer. Vi har nu føjet 2 NP-komplette problemer til vor liste, og en af dem havde endda noget at gøre med grafproblemer, som vi tidligere har været inde på. Man kan fortsætte herfra; faktisk kan vi bruge VC til at vise, at problemet om at finde ud af om en given graf har et Hamilton kredsløb, er NP-komplet. Det fører så ret umiddelbart til, at vort gennemgående eksempel, Traveling Salesman (i decisionsudgaven), også er det. Og der kan fortsættes til andre problemer, herunder velkendte problemer som heltalsprogrammering m.m.

9. Litteratur

En grundig indføring i emnet såvel som en meget omfattende liste af NP-komplette problemer, der samtidig kan bruges som en værktøjskasse ved eftervisning af NP-komplethed for nye problemtyper, kan findes i den tidligere nævnte grundbog på området, Garey og Johnson [1979].

KAPITEL 14

Punktprocesser og fornyelse

1. Stokastiske processer i operationsanalysen

I en lang række operationsanalytiske problemstillinger er det en væsentligt del af den foreliggende situation, at resultatet af de handlinger, man vælger at gennemføre, afhænger af udfaldet af en eller anden usikker hændelse. Når dette aspekt er tilstrækkelig vigtigt i den foreliggende sag, vil man vælge en model, hvor usikkerheden indgår som en eksplicit formuleret del af modellen. I de klassiske operationsanalytiske problemer af denne art betyder dette, at man vil anvende sandsynlighedsteoretiske overvejelser i sin modelopbygning og modelløsning.

Da vi i de følgende kapitler skal beskæftige os netop med den slags problemer, kan det ikke undgås, at vi må håndtere lidt sandsynlighedsregning undervejs. Egentlig er der ikke så meget nyt; imidlertid er det ofte en lidt anden slags overvejelser end dem, man kender f.eks. fra teoretisk statistik. Det er derfor værd at give dem et ord med på vejen, og det gør vi nu:

I sandsynlighedsregningen starter man med en eller anden grundlæggende tilfældighedsmekanisme. Hvad det er for en, behøver slet ikke at bekymre os, men vi kan antage, at der er givet en vis mængde X af mulige udfald; de enkelte udfald x fra X kan ikke forudsiges med sikkerhed, men der er alligevel en vis systematik, nemlig den, der fremkommer ved tilordning af sandsynligheder: For en delmængde A af X kan vi angive sandsynlingheden $P(A)$ for at det bliver udfald x fra A, som vælges. (På dette sted vil nogle vide, at man, når det går rigtigt til, skal være lidt påpasselige med, hvad det er for mængder A, man tager sandsynligheden af, de skal nemlig tilhøre en bestemt familie af delmængder af X, som man ofte skriver $\mathbf{X}$; hele herligheden $(X, \mathbf{X}, P)$ kaldes så et sandsynlighedsrum. Det kan vi godt tage afslappet i det følgende; det kommer ikke til at betyde så meget for os).

Vi kan nu introducere hovedpersonen i sandsynlighedsregningen, nemlig en såkaldt stokastisk variabel. Det er en funktion fra det underliggende rum af tilfældige udfald X til mængden af reelle tal. Hvis X var de enkelte små segmenter af en roulette, f.eks. den fra lykkehjulet, så er gevinsterne til deltagerne en stokastisk variabel. Vi skal se nok af dem senere hen (de stokastiske variable, altså). Stokastiske variable kan også tage heltallige værdier, og man kan have vektorer af stokastiske variable. (Den pertentlige og bedrevidende læser, som vi prøvede at

ryste af for lidt siden, vil vide, at det ikke er enhver funktion, der er en stokastiske variabel; den skal matche med den nævnte mængde $\mathbf{X}$ af tilladte delmængde, man siger at funktionen skal være målelig. Igen er det ikke noget, vi får brug for, hvilket ikke betyder, at det ikke er vigtigt; men de kvaler, man kan komme i ved at ignorere problemet, er checket, og de opstår ikke for os.)

Når man har en stokastisk variabel, der som nævnt blot er en funktion $f : X \to \mathbb{R}$, kan man se på sandsynligheden af mængder af typen $\{x \in X | f(x) \leq r\}$ for et vilkårligt tal r. Hvis vi kalder intervallet på talaksen fra $-\infty$ til og med r for $]-\infty, r]$, kan vi ligefrem opfatte sandsynligheden af disse halvåbne intervaller som en funktion F af r; vi skriver den som

$$F(r) = P\{f(x) \leq r\}$$

eller blot $P\{f \leq r\}$. Man vil nok nikke genkendende til F som fordelingsfunktionen for f. Hvis denne er differentiabel, kaldes dens afledede for tæthedsfunktionen for F.

Så skal vi bare have vort sidste generelle begreb, nemlig en stokastisk proces. En stokastisk proces er en familie $(f_t)_{t \in T}$ af stokastiske variable defineret på det samme under liggende sandsynlighedsrum, hvor T er en mængde af tidspunkter. T kan være $\{0, 1, 2, \ldots\}$ (diskret tid) eller $T = [0, \infty]$ (kontinuert tid) – eller noget helt tredie, dette sidste dog ikke hos os. Der er sådan set ikke noget mystisk over en stokastisk proces; grunden til, at der findes en hel teori om dem, er at de stokastiske variable (f_t) på forskellige tidspunkter t hører mere eller mindre tæt sammen.

Hvis der er valgt et givet udfald $x \in X$, så vil de stokastiske variables værdier $f_0(x), f_1(x), \ldots,$ (i diskret tid) eller generelt $f_t(x), t \in T$ beskrive et forløb over tid, en såkaldt trajektorie. Et af formålene med teorien om stokastiske processer er at fortælle, hvad der sker med sådanne trajektorier, efterhånden som tiden går. Det skal vi se anvendt på flere forskellige måder i det følgende. De stokastiske processer, som vi får med at gøre, er overvejende af en sådan art, at man kan sige noget om udviklingen over tid (selvom det ofte er svært at sige noget om de enkelte stokastiske variable).

2. Punktprocesser

Den type af stokastiske processer, som finder anvendelse i operationsanalysen, går under betegnelsen punktprocesser. Betegnelsen skriver sig fra en række anvendelser, hvor de begivenheder, der indtræffer, kan lokaliseres til bestemte områder (i rummet, planen eller på en linie), således at man kan spørge, hvor mange punkter i et givet område, som er opstået i et vist interval. F.eks. kan det, hvis man stiller sin bil på en parkeringsplads, have interesse at vide, hvor mange fugleklatter der vil være kommet på bilen i dagens løb. Dette er et oplagt emne for en beskrivelse ved en punktproces; at det ikke vil optræde i det følgende, skyldes kun, at andre lige så oplagte emner har større praktisk interesse.

Den mest prominente punktproces er *Poisson-processen*. Den kan bruges, som den er, og det gør vi ofte i det følgende, og den kan udbygges på forskellig måde. I sin simpleste form har vi at gøre med en bestemt begivenhed (f.eks. at en fugl i luftrummet over min bil smider en klat), som indtræder med en given *intensitet*, forstået således, at sandsynligheden for at det sker i et lille interval $\triangle t$ er $\lambda \triangle t$ (strengt taget skal dette kun gælde i grænsen, altså for $\triangle t$ gående mod nul). Givet denne antalelse kan sandsynligheden for k klatter i tidsrummet $[0, t]$ udregnes som

$$P\{N_t = k\} = e^{-\lambda t} \frac{(\lambda t)^k}{k!}.$$

Hvor er punkterne i denne proces? Jo, punkterne er på tidsaksen og markerer tidspunktet for klat nr. 1, nr. 2 osv. Strengt taget er historien også en smule mere omstændelig end i versionen ovenfor, for hvad vil det sige, at en bestemt begivenhed sker et antal gange? Begivenheden "fugleklat nr. 1" en faktisk en anden end "fugleklat nr. 7" (enhver bilejer vil ihvertfald vide, at den første ridse i lakken er en langt mere traumatisk oplevelse en ridse nr. 1000).

Vi vil i det store og hele klare os igennem uden for megen fordybelse i det formelle apparat, men på den anden side er det godt at være forsigtig med altfor naivistisk fremfærd i sandsynlighedsteoretiske problemer; det kan nemlig godt snyde, som vi skal se senere i kapitlet (i forbindelse med det såkaldte "busparadoks").

Vi indfører nu den stokastiske proces N_t (der altså er en funktion af den underliggende sandsynlighedsmekanisme *og* af tiden) defineret ved at

$$N_t = \text{største fodtegn } n \text{ så } T_n \leq t,$$

hvilket jo netop giver os antal begivenheder indtruffet op til tidspunkt t.

Nu virker denne definition jo ikke synderlig professionel, og der findes da også en smart måde at skrive det på. Vi skal bruge *den karakteristiske funktion:* Hvis vi lader A være en (tilladt) delmængde af det underliggende sandsynlighedsrum X, så kan vi definere den karakteristiske funktion af A som

$$1_A(x) = \begin{cases} 1 & \text{hvis } x \in A \\ 0 & \text{hvis } x \notin A \end{cases}$$

Dette er egentlig blot en device, der siger "ja" hvis x er i den undersøgte mængde A og "nej" ellers. Og så er det for resten en stokastisk variabel, en funktion på vort underliggende sandsynlighedsrum.

Med dette i hånden kan vi strømlinie udtrykket ovenfor noget, idet man jo kan skrive

$$N_t = \sum_{\nu=1}^{\infty} 1_{\{T_\nu \leq t\}}.$$

For hvert ν har vi taget mængden af de udfald, der fører til at T_ν er indtrådt på tidspunkt t. Når vi for et givet udfald summer antal gange, vi får "ja" ved at spørge

om udfaldet ligger i disse mængder, har vi netop antal begivenheder indtrådt op til tidspunkt t.

Man kan måske umiddelbart synes, at der ikke er opnået så meget ved at få en ny formel i stedet for den, man havde. Det er dog ikke helt rigtigt; charmen ved vor nye formel er, at den viser, hvorledes N_t kan skrives som en sum af stokastiske variable, nemlig $1_{T\nu \leq t}$ for $\nu = 1, 2 \ldots$, og en sådan fremstilling som en sum er ofte særdeles nyttig. Det opdager vi allerede i dette kapitel, når vi snakker om fornyelsesprocesser.

Inden vi forlader Poisson og fugleklatterne, vil vi notere en ret banal detalje: Vi kan selvfølgelig definere $N(]t_1, t_2]) = N_{t_2} - N_{t_1}$ for hvert (halvåbent interval $]t_1, t_2] = \{t | t_1 < t \leq t_2\}$; det giver os antallet af begivenheder i dette tidsinterval. Tilsvarende kan vi finde $N(A)$ for mere generelle delmængder af den positive akse; hvis A_1 og A_2 er sådanne delmængder, og de er disjunkte, vil der gælde $N(A_1 \cup A_2) = N(A_1) + N(A_2)$, og mere generelt har vi

$$N(\cup_{i=1}^{\infty} A_i) = \sum_{i=1}^{\infty} N(A_i)$$

for en vilkårlig familie af parvis disjunkte mængder $A_1, A_2, \ldots$ (med det sædvanlige forbehold, at mængderne skal være tilladte, "målelige"). Med andre ord, for hvert udfald af den underliggende tilfældighedsmekanisme giver processen et *mål* på den ikke-negative akse; det er iøvrigt et særdeles pænt og simpelt mål, nemlig et såkaldt tællemål, som for hver mængde tæller antallet af punkter fra en given familie hørende til mængden. Dette, som umiddelbart ser ud til at være et lidt pudsigt biprodukt, er faktisk fællestrækket ved alle punktprocesser, og det lægges til grund i den generelle definition. Denne generelle definition kan vi tage det roligt med; den er blot med for at antyde, at der er en fælles struktur i alle de processer, som vi ser på i de følgende praktiske anvendelser.

Helt generelt defineres en punktproces N på en mængde E (udstyret med en familie af tilladte delmængder $\mathcal{E}$) som en familie $\{N(A) | A \in \mathcal{E}\}$ af stokastiske variable på et underliggende sandsynlighedsrum, således at der for hvert udfald gælder, at N er et tællemål på E.

3. Genanskaffelsesproblemer

De foregående to afsnit har blot været definitioner. Vi går nu i gang med sagen, idet vi i dette kapitel vil se på en speciel type punktprocesser. Vi starter med en intuitiv behandling af nogle problemer, der har at gøre med at fornyelse; senere sætter vi stokastik på.

En virksomhed har normalt et fast anlæg bestående af enkelte maskiner, køretøjer m.m., der i forbindelse med virksomhedens drift er udsat for et vist slid

og derfor efter en kortere eller længere periode må erstattes med andre, tilsvarende eller lignende, genstande. Dette er velkendt og ligetil, men der er alligevel forskel på at vide, at en maskine må udskiftes på et eller andet tidspunkt – f.eks. når den bliver helt ubrugelig – og at udskifte den på *det rette* tidspunkt. Det er betragtninger omkring det sidste aspekt, som skal optage os her.

Vi starter med det *deterministiske* genanskaffelsesproblem. Først må det naturligvis præciseres, hvad der skal forstås ved det rette tidspunkt. Her er det sædvane at antage, at der er to typer af omkostninger, nemlig

• *værditab* opstået ved, at maskinens salgsværdi formindskes med tiden,

• *driftsomkostninger* forbundet med at holde maskinen i en sådan stand, at den kan udfylde sit formål.

Betegnes maskinens nypris med C, dens salgsværdi ved udløbet af den $k$te periode med $S(k)$, og er R_i de samlede driftsudgifter i den i'te driftsperiode (som antages betalt ved periodens start), har vi samlede tilbagediskonterede udgifter ved salg efter k perioder som

$$P = C - \beta^k S_k + \sum_{i=0}^{k-1} \beta^i R_i,$$

idet β er diskonteringsfaktoren.

Såfremt genanskaffelsespolitikken går ud på at sælge efter k perioder, fås tilhørende omkostninger pr. periode som den annuitet over k perioder, der tilbagediskonteret giver P, altså som løsningen x til

$$P = x + \beta x + \ldots + \beta^{k-1} x = x \frac{1 - \beta^k}{1 - \beta}.$$

Ved at indsætte udtrykket for P får vi, at den gennemsnitlige årlige omkostning ved k-periode politikken bliver

$$x = \frac{(C - \beta^k S_k + \sum_{i=1}^{k-1} \beta^i R_i)(1 - \beta)}{1 - \beta^k}.$$

For at finde den optimale politik skal vi nu minimere x som funktion af k. Rent praktisk vil det ofte – ihvertfald når problemet ikke er altfor stort – være nemmest at gøre det direkte, dvs. ved at gennemregne situationen for et passende antal værdier af k.

I den situation, vi har betragtet ovenfor, var essensen i problemet en afvejning af udgiften til nyt udstyr overfor omkostningerne ved at holde det gamle i funktionsduelig stand. Denne afvejning er relevant i mange, men ikke i alle sammenhænge. Et andet udskiftningsproblem opstår i tilfælde, hvor den enkelte genstand ikke – selv med store omkostninger – kan holdes i funktionsduelig stand. Ofte er der

endda tale om ganske små detaljer (en chips, en reservedel i en flymotor), der i nyanskaffelse ikke koster særlig meget, men hvor en defekt kan få vidtrækkende konsekvenser. På den anden side kan man heller ikke udskifte alle dele efter hver enkelt anvendelse.

Vi starter med en intuitiv tilgang. Antag, at vi har at gøre med en reservedel, som virksomheden bruger mange af, men hvor virkningen af et nedbrud er betydelig. Vi går ud fra, at virksomheden kender sandsynligheden for defekt i hver periode t, givet at der ikke har været fejl i de foregående $t-1$ perioder. Denne sandsynlighed betegnes med p_t.

Antag nu, at virksomheden udskifter reservedelen efter nøjagtig k perioder. Vi kan da finde den gennemsnitlige levealder θ_k af reservedelen (den afhænger jo *både* af, hvor tit der udskiftes *og* hvor mange nedbrud der kommer) som

$$\theta_k = p_1 + 2p_2 + 3p_3 + \cdots + (k-1)p_{k-1} + k(p_k + p_{k+1} + \cdots).$$

Lader vi sandsynligheden for nedbrud efter år t være

$$P_t = p_{t+1} + p_{t+2} + \cdots,$$

har vi, at

$$\theta_k = P_0 + P_1 + \cdots + P_{k-1}.$$

Er antallet af perioder, vi betragter, et stort tal T, skal vi bruge T/θ_k reservedele. Da vi udskifter efter k perioder, vil der i hver k perioder

$$\frac{T}{\theta_k}(p_1 + \cdots + p_{k-1}) = \frac{T}{\theta_k}(1 - P_{k-1}),$$

som går i stykker, og resten udskiftes efter k perioder. Vi antager, at omkostningerne er A ved at skifte reservedelen før tiden, og B er omkostningerne ved at skifte den til tiden; det ligger i problemstillingen, at A er væsentlig større end B. Ialt bliver omkostningerne da over de T perioder $(AT(1-P_{k-1}) + BTP_{k-1})/\theta_k$, hvilket svarer til omkostninger pr. periode på

$$C_k = \frac{A(1-P_{k-1}) + BP_{k-1}}{\theta_k} = \frac{A - (A-B)P_{k-1}}{\theta_k}.$$

For at vælge optimal udskiftningspolitik skal virksomheden nu tilpasse k, så at C_k bliver mindst muligt.

Behandlingen af dette problem kan udbygges på flere måder, og det er det også blevet i litteraturen. Problemet om at udskifte reservedele er vokset siden den spæde barndom, som er refereret i det foregående, og blevet til en selvstændig operationsanalytisk disciplin, som vi vil diskutere i næste kapitel. Her vil vi gå lidt dybere ned i den sandsynlighedsteoretiske baggrund, således at vi kan erstatte de

intuitive argumenter (f.eks. om at man bruger T/θ_k dele, når T er stor, og derefter forkorter T væk) med lidt mere præcise. For det andet skal vi også trække et par andre perspektiver ind. Det vil resten af kapitlet handle om.

4. Fornyelsesprocesser; Blackwell's sætning

I dette afsnit fortsætter vi med problemstillingen fra det forrige afsnit, idet vi dog formulerer det lidt anderledes for at kunne trække på en række sandsynlighedsteoretiske resultater.

Lad os starte med at kigge på en enkelt maskine eller maskintype. Vi vil antage, at der, når maskinen går i stykker, enten foretages en reparation eller en udskiftning, men under alle omstændigheder, at maskinen efter reparationen (i en vis forstand) er som ny.

I terminologien fra afsnit 1 vil vi arbejde med en følge $(\xi_j)_{j=1}^{\infty}$ af stokastiske variable, hvor ξ_j er den tid der går, inden den j'te maskine (den maskine, man har efter $j-1$ nyanskaffelser eller reparationer) går i stykker (bemærk, at vi har en stokastisk proces i diskret tid; at værdierne af de enkelte stokastiske variable kan fortolkes som "tid" (inden nedbrud), er egentlig blot en pudsighed). Antagelsen ovenfor om, at maskinerne efter hver nyanskaffelse eller reparation starter på en frisk, går ud på at de stokastiske variable ξ_j er *stokastisk uafhængige*. Videre antages det normalt, at alle variablene ξ_j for $j \geq 2$ har samme fordeling; øvrigt behøver vi i første omgang ikke antage andet om fordelingen end at der er endelig og positiv middelværdi,

$$\mathsf{E}\xi_j = a > 0, \text{ alle } j \geq 2.$$

Den første variable ξ_1, kan godt have en anden fordeling, svarende til, at en ny maskine er kvalitativt anderledes end den maskine, der har været gennem nogle reparationer.

Givet denne underliggende mekanisme for maskinernes nedbrud, kan vi definere den tilhørende *fornyelsesproces* som en familie $\{\eta(t)|t \geq 0\}$ af stokastiske variable $\eta(t)$, hvor parameteren t gennemløber alle positive tal, idet

$$\eta(t) = \max\{k|S_k \leq t\},\ t \geq 0,$$

hvor $S_k = \sum_{j=1}^{k} \xi_j$,, og $S_0 = 0$. Her er S_k tydeligt nok den samlede tid, der går inden maskinen er brudt ned ialt k gange, og $\eta(t)$ bliver dermed til det antal gange, maskinen har været nede inden tidspunkt t.

(Teknisk set er der den lille detalje, at maskinen i princippet kan gå i stykker uendelig ofte inden tidspunkt t – f.eks. første gang til tidspunkt $t(1-2^{-1})$, k'te gang til tidspunkt $t(1-2^{-k})$, så at der slet ikke er noget maximum i formlen. I så fald sætter vi $\eta(t)$ til $+\infty$.)

Det er ofte mere praktisk at se familien $\{\eta(t)|t \geq 0\}$ som en *stokastisk proces,* dvs. en tilfældighedsmekanisme, der resulterer i funktioner η på den ikke-negative

talakse. Det er endda en punktproces, idet der for enhver (målelig) delmængde A at den ikke-negative talakse gælder, at $\eta(A)$ (= antal nedbrud i det tidsrum, der beskrives af A) er en stokastisk variabel (en funktion af den underliggende tilfældighedsmekanisme) og at der for hvert givet udfald gælder, at $\eta(\cdot)$ er additiv på delmængder af tidsaksen.

Her skal vi især interessere os for en ganske almindelig (dvs. deterministisk) funktion af tiden t, som kan defineres udfra $\{\eta(t)|t \geq 0\}$, nemlig

$$H(t) = \mathsf{E}\eta(t).$$

Vi kan fortolke $H(t)$ som det gennemsnitlige antal nedbrud op til tidspunkt t.

Vi kan (jvf. afsnit 2) skrive $\eta(t)$ på en anden måde ved at indføre *indikatorfunktionerne* $1_{\{S_j \leq t\}}$ for $j = 1, \ldots, k$. Disse indikatorfunktioner er definerede på det underliggende sandsynlighedsrum, og er givet ved

$$1_{\{S_j \leq t\}} = \begin{cases} 1 & \text{hvis } S_j \leq t \\ 0 & \text{ellers} \end{cases}$$

Vi har da, at

$$\eta(t) = \sum_{j=1}^{\infty} 1_{\{S_j \leq t\}}.$$

Fordelen ved denne skrivemåde kommer frem, når vi tager middelværdi: For det første kan middelværdien af summen skrives som summen af middelværdier, og for det andet lader middelværdien af en indikatorfunktion sig beregne meget let, nemlig ved

$$\mathsf{E}1_{\{S_j \leq t\}} = P\{S_j \leq t\},$$

så at vi ialt har

$$H(t) = \sum_{j=1}^{\infty} P\{S_j \leq t\}.$$

For at simplificere sagen vil vi i det følgende antage, at ξ_j'erne kun antager positive heltalsværdier, altså f.eks. 1,2,3, osv. (det svarer til at vi har valgt visse mindste tidsenheder i vor tidsmåling).

Vi starter med at se på et nyttigt specialtilfælde, nemlig de såkaldte *homogene* fornyelsesprocesser: Vi skriver ξ_j's fordeling (for alle $j \geq 2$, de har jo samme fordeling) som $(p_k)_{k=1}^{\infty}$ med $\sum_k p_k = 1$, og tilsvarende ξ_1's fordeling som $(q_k)_{k=1}^{\infty}$. Processen $\{\eta(t)|t \geq 0\}$ kaldes homogen hvis der gælder,at

$$q_k = \frac{1}{a} \sum_{j=k}^{\infty} p_j, \; k = 1, 2, \ldots.$$

Bemærk, at hvis vi summerer over alle k, dvs. laver summen

$$\sum_{k=1}^{\infty} q_k = \frac{1}{a} \sum_{k=1}^{\infty} \sum_{j=k}^{\infty} p_j,$$

skal vi i dobbeltsummen først tage p_1 en gang, dernæst p_2 to gange, så p_3 tre gange, osv. Men det svarer netop til at udregne $\mathsf{E}\xi_j = a$, så summen af $q_k$erne bliver 1.

Homogene fornyelsesprocesser giver en simpel funktion H:

Lad $\{\eta(t)|t \geq 0\}$ være homogen. Da gælder, at

$$H(k) = \frac{k}{a}.$$

Da vi skal bruge formlen for $H(k)$ ovenfor, altså sandsynligheder for summer af uafhængige stokastiske variable, er det praktisk at bruge frembringende funktioner. (Umiddelbart har man eventuelt det handicap, at man aldrig har hørt om sådan en før; det viser sig dog at være til at overkomme, for frembringende funktioner er ret brugervenlige: Ideen er at lade den givne sandsynlighedsfordeling repræsentere som koefficienterne i en potensrække i en eller anden abstrakt variabel z; charmen ved denne tilsyneladende besværlige omskrivning er at man så får fordeling af summer af uafhænginge stokastiske variable ved at gange potensrækker sammen, og det kan man med fordel spille på, bare se:)

Vi starter med den frembringende funktion for ξ_2 (eller for ξ_j for $j \geq 2$, som er den samme); kaldes den for $p(z)$, har vi

$$p(z) = \mathsf{E}z^{\xi_2} = \sum_{k=1}^{\infty} p_k z^k.$$

Den frembringende funktion $q(z)$ for ξ_1 bliver

$$\begin{aligned} q(z) =& \frac{1}{a} \sum_{k=1}^{\infty} \sum_{j=k}^{\infty} p_j z^k \\ =& \frac{z}{a} \sum_{j=1}^{\infty} p_j \sum_{k=0}^{j-1} z^k \\ =& \frac{z}{a} \sum_{j=1}^{\infty} p_j \frac{1-z^j}{1-z} = \frac{z(1-p(z))}{a(1-z)}. \end{aligned}$$

Hvis vi endelig indfører betegnelsen $\psi(z)$ for den frembringende funktion for følgen $(h(k))_{k=1}^{\infty}$ givet ved $h(k) = H(k) - H(k-1)$, her vi

$$\begin{aligned}\psi(z) &= \sum_{k=1} z^k h(k) = \sum_{j=1}^{\infty}\sum_{k=1}^{\infty} z^k P\{S_j = k\} \\ &= \sum_{j=1}^{\infty} \mathsf{E} z^{S_j} = q(z)\sum_{j=1}^{\infty}[p(z)]^j \\ &= \frac{q(z)}{1-p(z)} = \frac{z}{a(1-z)}.\end{aligned}$$

Sammenlignes nu udtrykkene for $\psi(z)$ i starten og slutningen af denne udregning, ses, at alle $h(k)$ må være ens og lig med $1/a$. □

Foruden størrelsen $\eta(t)$ er det bekvemt også at arbejde med $\nu(t)$ givet ved

$$\nu(t) = \eta(t) + 1.$$

Det ses fra definitionerne, at der gælder

$$\nu(t) = \min\{k | S_k > t\},$$

således at $\nu(t)$ er den stokastiske variable, der på tidspunkt t angiver nummeret på den maskine, der følger efter næste nedbrud. Fortolkningen af $\nu(t)$ er mindre intuitiv end for $\eta(t)$, og det er da heller ikke af hensyn til denne fortolkning, at man inddrager $\nu(t)$; den er interessant fordi det for ethvert n kan det udfra observation af variablene $\xi_1, \ldots, \xi_n$ afgøres, om $\nu(t) \leq n$. Det samme er ikke tilfældet for $\eta(t)$, for hvis vi kun ved noget om de første n variable $\xi_1, \ldots, \xi_n$, kan vi ikke afgøre, om ikke endnu en når at bryde ned inden t.

Vort første centrale resultat i denne model har at gøre med det gennemsnitlige antal nedbrud af en maskine i et tidsinterval:

Lad $h(k) = H(k) - H(k-1)$. *Da gælder*

$$h(k) \to \frac{1}{a} \text{ } \textit{og} \text{ } H(k) \sim \frac{k}{a} \text{ } \textit{for} \text{ } k \to \infty.$$

Umiddelbart kan man måske synes, at dette resultat (den såkaldte Blackwell's sætning, der regnes for central i teorien om fornyelsesprocesser) ikke er noget at råbe hurra for. Når den gennemsnitlige levetid af en maskine er a små tidsintervaller, vil der i løbet af et sådant tidsinterval gennemsnitligt være $1/a$ maskiner, der går ned. Det er dog ikke oplagt (og iøvrigt også kun rigtigt efter at der er gået tilstrækkelig

lang tid), så det kræver et argument; dertil kommer, at man med dette resultat kan komme lidt længere og finde *fordelingerne* snarere end blot middelværdierne.

For at bevise Blackwell's sætning betragter vi to følger af stokastiske variable $(\xi_j)_{j=1}^{\infty}$ og $(\xi'_j)_{j=1}^{\infty}$; variablene i de to følger er uafhængige, ξ_1 har vilkårlig fordeling, mens ξ_j og ξ'_j har samme fordeling for alle j med $P(\{\xi_j = k\}) = P(\{\xi'_j = k\}) = p_k$ for alle k og $j > 2$. Fordelingen af ξ'_1 er bestemt ved

$$q_k = P\{\xi_1 = k\} = \frac{1}{a}\sum_{j=k}^{\infty} p_j$$

(så den tilhørende $\eta'(t)$ er en af de homogene fornyelsesproces behandlet ovenfor). Summerne af ξ_j og ξ'_j betegnes med S_n og S'_n.

Lad μ være den stokastiske variable givet ved

$$\mu = \min\{n \geq 1 | S_n = S'_n\}.$$

Da kan det for hvert n udfra observation af $\xi_1, \ldots, \xi_n$ og $\xi'_1, \ldots, \xi'_n)$ afgøres, om $\mu \leq n$. Det betyder igen, at variablene i denne følge op til μ, og variablene efter μ vil være uafhængige, fordelingen af variablene $(\xi_{\mu+1}, \xi_{\mu+2}, \ldots)$ er den samme som $(\xi_2, \xi_3, \ldots)$, og tilsvarende for ξ'_j. Men så kan vi erstatte $(\xi_{\mu+1}, \xi_{\mu+2}, \ldots)$ i summen S_n med $(\xi'_{\mu+1}, \xi'_{\mu+2}, \ldots)$ uden at ændre på fordelingen af S_n.

Betragter vi nu mængden $\{S_\mu < k\}$ af alle de udfald, hvor de to følger bliver ens inden der er gået k tidsenheder, har vi $\eta(t) = \eta'(t)$ for $t \geq k-1$, og derfor får vi

$$\begin{aligned} h(k) &= \mathsf{E}\left[\eta(k) - \eta(k-1)\right] = \\ &= \mathsf{E}\left[(\eta'(k) - \eta'(k-1))1_{\{S_\mu < k\}}\right] \\ &\qquad + \mathsf{E}\left[(\eta'(k) - \eta'(k-1))1_{\{S_\mu \geq k\}}\right] \\ &= \frac{1}{a} - \mathsf{E}\left[(\eta'(k) - \eta'(k-1))1_{\{S_\mu \geq k\}}\right] \\ &\qquad + \mathsf{E}\left[(\eta(k) - \eta(k-1))1_{\{S_\mu \geq k\}}\right]. \end{aligned}$$

Vi har dermed, at $h(k)$ afviger fra $1/a$ med en størrelse, der er givet ved de to sidste led i summen. Vi har, at

$$|\eta(k) - \eta(k-1)| \leq 1$$

(dette er fordelen ved at arbejde med diskret tid – der må gå mindst én tidsenhed fra et nedbrud til det næste), kan hver af de to middelværdier ikke blive større end sandsynligheden for at S_μ er større eller lig k, så

$$\left| h(k) - \frac{1}{a} \right| \leq 2P\{S_\mu \geq k\} \to 0$$

for $k \to \infty$. Dette er første del af sætningen. Anden del følger umiddelbart. □

Vort resultat er udledt under den antagelse, at de stokastiske variable ξ_j kun tager heltalsværdier (dvs. at vi har diskret tid). Det viser sig hurtigt at være ubekvemt i anvendelserne, men heldigvis holder Blackwells sætning generelt. Beviset for det kan i det store og hele køres på samme måde, men for at gøre det må man have lidt mere værktøj, og det vil vi ikke komme ind på.

5. Den asymptotiske fordeling; busparadokset

Fra vor fornyelsesproces $\{\eta(t)|t \geq 0\}$ kan vi definere to nye stokastiske variable, nemlig *excessen*

$$\chi(t) = S_{\nu(t)} - t > 0$$

og *defekten*

$$\gamma(t) = t - S_{\eta(t)} \geq 0.$$

Vi kan fortolke $\chi(t)$ som den restperiode, som maskinen stadig vil fungere i på tidspunkt t; omvendt er $\gamma(t)$ den tid, som maskinen på tidspunkt t allerede har fungeret.

Umiddelbart kunne man tro, at $\chi(t) + \gamma(t)$, som er samlet funktionstid af den maskine, man har på tidspunkt t, har samme fordeling som ξ_j for $j \geq 2$, men det passer ikke. Specielt kan man udmærket komme ud for den situation, at

$$\mathsf{E}\chi(t) > \mathsf{E}\xi_j = a$$

for passende store t.

Dette kan iøvrigt formuleres som et af sandsynlighedsregningens mange tilsyneladende paradokser (der dog passer fint med ens egen private empiri): En buspassager, som på tidspunkt t ankommer til et stoppested, hvor bussen ankommer med intervaller $\xi_1, \xi_2, \ldots$ (med $\mathsf{E}\xi_j = a$ for $j \geq 2$; tallet a fremgår af køreplanen på stoppestedet) vil typisk vente ulideligt længe på bussen; det kan faktisk forekomme, at den *gennemsnitlige* ventetid er længere end a!

Vi skal vende tilbage til busparadokset. Det væsentlige ved vor generelle situation er, at den simultane fordeling af $\chi(t)$ og $\gamma(t)$ for voksende t går mod en asymptotisk sandsynlighedsfordeling. Det fremgår af følgende sætning:

Lad processen $\{\eta(t)|t \geq 0\}$ være givet, hvor ξ_j for $j > 1$ er identisk fordelte med $\mathsf{E}\xi_2 = a$, og ξ_1 har en vilkårlig fordeling. Da gælder, at

$$P\{\gamma(k) = i, \chi(k) = j\} \to \frac{p_{i+j}}{a}$$

for $k \to \infty$.

Vi har

$$\begin{aligned} P\{\gamma(k)=i,\chi(k)=j\} =& \sum_{h=1}^{\infty} P\{S_h = k-i, \xi_{h+1} = i+j\} \\ =& \sum_{h=1}^{k-i} P\{S_h = k-i\} P\{\xi_2 = i+j\} \\ =& h(k-i)p_{i+j} \to \frac{p_{i+j}}{a} \end{aligned}.$$

hvor vi har brugt Blackwell's sætning til sidst. □

Vi er intersserede i den asymptotiske fordeling for $\chi(t)$. Vi har

$$\lim_{k\to\infty} P\{\chi(k)=i\} = \frac{1}{a}\sum_{j=i}^{\infty} p_j,$$

dvs. samme fordeling som ξ_1 i en homogen fornyelsesproces. Faktisk svarer det til at det allerede fra processens start virker som om den har fungeret i en længere periode – og det er også derfor, at man bruger betegnelsen homogen om sådanne processer. Er processen *ikke* homogen, vil $\chi(t)$ have en fordeling der afhænger af t, idet den første variabels effekt kun lidt efter lidt forsvinder.

Generelt kan vi udregne $\mathsf{E}\chi$, middelværdien af den asymptotiske fordeling for $\chi(t)$, til

$$\begin{aligned} \mathsf{E}\chi =& \frac{1}{a}\sum_{k=1}^{\infty} k(p_k + p_{k+1} + \cdots) \\ =& \frac{1}{a}\sum_{j=1}^{\infty} p_j(1 + \cdots + j) \\ =& \frac{1}{a}\sum_{j=1}^{\infty} p_j \frac{j(j+1)}{2} = \frac{1}{2a}\mathsf{E}\xi_2^2 + \frac{1}{2a}\mathsf{E}\xi_2 = \frac{1}{2a}\mathsf{E}\xi_2^2 + \frac{1}{2}. \end{aligned}$$

Det viser os, at hvis ξ_2 er valgt således, at

$$\mathsf{E}\xi_2^2 > 2a^2 - a,$$

da vil $\mathsf{E}\chi(n) > \mathsf{E}\xi_j = a$ for n tilstrækkelig stor (busparadokset).

6. Maskinproblemet med 1 operatør

Vi vender os nu til en mere praktisk orienteret udgave af fornyelsesproblemet; vi antager, at der i en virksomhed er m maskiner af samme type, som fra tid til anden går i stykker og må repareres. Der er et vist antal medarbejdere, operatører, hvis job det er at reparere maskinerne; imidlertid kan det tænkes, at der går flere maskiner i stykker, mens de tidligere repareres, og som følge heraf kan et større eller mindre antal maskiner være midlertidig ude af drift.

Vi er interesserede i at bestemme den gennemsnitlige reparationstid for maskinerne. Det har naturligvis betydning for tilrettelæggelsen af driften, at denne reparationstid gøres så lille som mulig, naturligvis under behørig hensyn til omkostningerne herved. Den gennemsnitlige reparationstid kan påvirkes ved at der indsættes flere operatører, eller ved at maskinernes tekniske parametre ændres.

Vi skal i første omgang især interessere os for, hvorledes den gennemsnitlige reparationstid kan findes. Vi vil derfor ikke komme ind på de samlede stilstandsomkostningers afhængighed af antallet af operatører, men hele vejen antage, at der kun er en enkelt.

Klarperioder og reparationsperioder. I vor repræsentation af maskinproblemet vil vi antage, at der er m maskiner, som hver arbejder i visse perioder $w_1, w_2, \ldots$ med en stilstandsperiode imellem. Vi antager, at hver af arbejdsperioderne er uafhængige stokastiske variable, som er eksponentialt fordelt, således at

$$P(\{w_i \leq t\}) = 1 - e^{-\lambda t},$$

hvor λ er en konstant. Anvender vi nu teorien fra sidste afsnit på de stokastiske variable $w_1, w_2, w_3, \ldots$, ser vi, at

$$\mathsf{E}w_i = \frac{1}{\lambda},$$

så $\lambda = H(1)$ er det gennemsnitlige antal nedbrud pr. maskintime; omvendt er $1/\lambda = 1/H(1)$ den gennemsnitlige varighed af en arbejdsperiode (strengt taget har vi i forrige afsnit kun vist dette for diskret tid, men det holder som nævnt også når tiden varierer kontinuert; iøvrigt følger det i vort specielle tilfælde fra sammenhængen mellem eksponential- og Poissonfordelingen).

I første omgang vil vi antage, at reparationstiden u for en maskine følger en diskret fordeling med

$$P(\{u = u_i\}) = \theta_i$$

for et vist antal mulige reparationstider $u_1, u_2, u_3, \ldots$ (som man kan forestille sig som 1,2,3 osv. arbejdsdage); den gennemsnitlige reparationstid bliver da

$$\mathsf{E}u = \sum_{i=1}^{\infty} u_i \theta_i,$$

og tilsvarende får vi det gennemsnitlige antal fuldførte reparationer pr. tidsenhed som

$$\mu = \frac{1}{\mathsf{E}u}.$$

Fra starten er alle m maskiner igang; dette kaldes for en *klarperiode*. Derpå går den første i stykker og kommer under reparation, hvorved en *reparationsperiode* påbegyndes. Mens reparationen af den første maskine står på, kan den næste maskine tænkes at bryde ned, og den må da afvente, at operatøren har færdiggjort reparationen af nummer 1. Således fortsættes, og en reparationsperioden er først endt, når der ved afslutningen af en reparation *ikke venter* en nedbrudt maskine.

Vi betegner den gennemsnitlige længde af en reparationsperiode med x_m, og det gennemsnitlige antal maskintimer, som gennemføres under en reparationsperiode, med v_m. Tilsvarende betegnes med x'_m og v'_m gennemsnitlig varighed og gennemsnitligt antal præsterede maskintimer under klarperioden. Vi har

$$v'_m = mx'_m$$

idet samtlige maskiner jo er i gang under en klarperiode. Da en klarperiode ender med et nedbrud, har vi

$$\lambda v'_m = 1,$$

så at vi i det følgende kan erstatte v'_m med $1/\lambda$ og x'_m med $1/m\lambda$.

Ser vi dernæst på en reparationsperiode, har vi, at den gennemsnitlige reparationstid for en maskine er $1/\mu$, så det gennemsnitlige antal reparationer i en reparationsperiode er μx_m. Dette antal må være en større end antallet af nedbrud (for det første nedbrud, som startede reparationsperioden, regnes ikke med, og hele perioden ender med, at der ikke er flere maskiner, som er brudt ned), altså

$$\mu x_m = 1 + \lambda v_m.$$

Hermed har vi, at alle størrelser kan bestemmes, når vi kender x_m (samt naturligvis parametrene μ og λ).

Bestemmelse af x_m. Det er ikke helt ligetil at få et udtryk for x_m, så vi bliver nødt til at gå en omvej.

1. skridt: Vi starter med sandsynligheden for, givet at der på et vist tidspunkt er s maskiner igang, at der τ tidsenheder senere vil være r af dem, som stadig er igang. Det svarer til, at $s - r$ maskiner må være stoppet i løbet af intervallet τ, og da sandsynligheden for et stop i løbet af τ er $1 - e^{-\lambda\tau}$, får vi den ønskede sandsynlighed som

$$p_{s,r}(\tau) = \binom{s}{r} e^{-r\lambda\tau}(1 - e^{-\lambda\tau})^{s-r}.$$

2. skridt: Vi betragter nu en reparationsperiode, om hvilken vi foreløbig antager, at første reparation tager netop tiden u_i. Når denne første reparation er forbi, kan der være et vilkårligt antal maskiner $r+1$ med r gående fra 0 til $m-1$ i gang (idet den netop reparerede regnes med). Sandsynligheden for, at der netop er $r+1$ igang er vor $p_{m-1,r}(u_i)$ fra før. Hvis $r = m-1$ ender reparationsperioden efter u_i, dvs. $x_m = u_i$. Hvis $r = m-2$, er der netop én maskine, som afventer reparation, og situationen er exakt som ved starten af en reparationsperiode, dvs. der kræves gennemsnitlig endnu en reparationstid på x_m.

Vi fortsætter nu med at se på tilfældet $r = m-3$. Når den første reparation er færdig, vil der være to maskiner, der venter. Operatøren går i gang med den ene, mens den anden får lov at vente; det svarer til, at operatøren passer $m-1$ (i stedet for egentlig m) maskiner, mens en maskine er helt udenfor. Den gennemsnitlige reparationstid i denne situation er x_{m-1}, hvorefter alle $m-1$ maskiner er i gang, og operatøren kan gå over til den maskine, der midlertidig var sat udenfor. Der går nu gennemsnitlig x_m inden alle maskiner er igang. Ialt har vi at for $r = m-3$ kræves i gennemsnit $x_{m-1} + x_m$ efter første reparation indtil reparationsperioden er slut.

Går man de tilsvarende argumenter igennem for mindre værdier af r, får man, at hvis den første reparation varer u_i, er den gennemsnitlige længde

$$\begin{aligned} u_i + p_{m-1,m-2}(u_i)x_m + p_{m-1,m-3}(u_i)(x_m + x_{m-1}) + \cdots \\ + p_{m-1,0}(x_m + \cdots + x_2) = \\ = u_i + \sum_{r=0}^{m-2} (x_m + \cdots + x_{r+2}) p_{m-1,r}(u_i), \end{aligned}$$

som, når vi indsætter for $p_{m-1,r}(u_i)$, bliver til

$$u_i + \sum_{r=0}^{m-2} \binom{m-1}{r} e^{-r\lambda u_i} (1 - e^{-\lambda u_i})^{m-r-1} (x_m + \cdots + x_{r+2}) \theta_i.$$

Vi mangler nu blot at tage gennemsnit over den første reparations længde, og vi får da

$$x_m = \frac{1}{\mu} + \sum_{r=0}^{m-2} \sum_{i=1}^{\infty} \binom{m-1}{r} e^{-r\lambda u_i} (1 - e^{-\lambda u_i})^{m-r-1} (x_m + \cdots + x_{r+2}).$$

Dette giver os et udtryk for x_m som funktion af alle x_r for $r < m$, og der kan, ihvertfald i princippet, løses for x_m. I første omgang vil vi gerne have alle led med x_m over på venstre side; på højre side af lighedstegnet er koefficienten til x_m

$$\sum_{i=1}^{\infty} \theta_i \sum_{r=0}^{m-2} \binom{m-1}{r} e^{-r\lambda u_i} (1 - e^{-\lambda u_i})^{m-r-1}.$$

Hvis den sidste sum havde gået fra 0 til $m-1$ i stedet for blot til $m-2$, havde denne sum været 1 (binomialformlen). Men så kan vi jo erstatte summem med $1-(m-1)$'te led, så at vi får

$$\sum_{i=1}^{\infty}[1-e^{-(m-1)\lambda u_i}]\theta_i.$$

Her kan vi gange ind under summetegnet og summe hvert led for sig, hvor vi udnytter at $\sum_{i=1}^{\infty}\theta_i = 1$. Når vi indsætter alt, får vi

$$x_m = x_m - A_m x_m + B_m,$$

eller

$$x_m = \frac{B_m}{A_m},$$

hvor

$$A_m = \sum_{i=1}^{\infty}\theta_i e^{-(m-1)\lambda u_i},$$

og

$$B_m = \frac{1}{\mu} + \sum_{r=0}^{m-3}\sum_{i=1}^{\infty}\binom{m-1}{r}e^{-r\lambda u_i}(1-e^{-\lambda u_i})^{m-r-1}(x_{m-1}+\cdots+x_{r+2})\theta_i.$$

Man kan nu gå i gang med at bestemme x_m nedefra: Vi får

$$x_1 = \frac{1}{\mu},\ x_2 = \frac{1}{\mu\alpha_1},\ x_3 = \frac{1}{\mu}\left(\frac{1}{\alpha_1\alpha_2}+\frac{1}{\alpha_1}-\frac{1}{\alpha_2}\right),$$

hvor α_k for $k = 1, 2, \ldots$ er bestemt ved

$$\alpha_k = \sum_{i=1}^{\infty}\theta_i e^{-k\lambda u_i}.$$

For højere m-værdier bliver regningerne ret besværlige.

I et særligt tilfælde kan vi simplificere udtrykket noget mere; antag at reparationerne alle varer lige længe (således at u_i er konstant med værdien $1/\mu$). Vi indfører den såkaldte *betjeningsfaktor*

$$\rho = \frac{\lambda}{\mu}$$

(der generelt udtrykker forholdet mellem gennemsnitligt antal nedbrud og gennemsnitligt antal reparationer pr. tidsenhed), og vi bruger nye variable $y_m = \mu x_m$

(eller, alternativt, måler tiden i sådanne enheder, at en reparation tager præcis en tidsenhed). Vi har da

$$e^{-(m-1)\rho}y_m = 1 + \sum_{r=0}^{m-3} \binom{m-1}{r} e^{-r\rho}(1-e^{-\rho})^{m-r-1}(y_{m-1} + \cdots + y_{r+2}),$$

eller, hvis vi for overskuelighedens skyld indfører $\nu = e^{\rho}$,

$$\nu^{-(m-1)}y_m = 1 + \sum_{r=0}^{m-3} \binom{m-1}{r} \nu^{-r}(1-\nu^{-1})^{m-r-1}(y_{m-1} + \cdots + y_{r+2}).$$

Dette udtryk kan løses for y_m, hvorved man får

$$y_m = 1 + \binom{m-1}{1}(\nu-1) + \binom{m-1}{2}(\nu-1)(\nu^2-1) + \cdots$$
$$\cdots + \binom{m-1}{m-1}(\nu-1)(\nu^2-1)\ldots(\nu^{m-1}-1).$$

Vi udelader beviset for, at dette faktisk er løsningen (et bevis for formlen, der skyldes Ashcroft (1950), kan findes hos denne). For små værdier af m kan man løse og finde

$$y_1 = 1,\ y_2 = \nu,\ y_3 = \nu^3 - \nu^2 + \nu.$$

For større m må man benytte formlen.

Anvendelser. Indtil nu har vi ikke set nogen anvendelser af vore overvejelser. De kommer ind, når vi ønsker at opnå bedst mulige arbejdsbetingelser for vort system (bestående af m maskiner og 1 operatør). Vi indfører følgende notation: Lad a_m, b_m og c_m være det gennemsnitlige antal af maskiner, som henholdsvis arbejder, repareres og afventer reparation. De økonomiske karakteristika af systemet med m maskiner og 1 operatør kan da opsummeres ved

- maskineffektivitet $\eta_m = a_m/m$,
- operatøreffektivitet b_m
- ventetidstab $l = c_m/m$.

Maskineffektiviteten er et mål for, hvor stor en andel af maskinparken som er igang; tilsvarende er operatøreffektiviteten et mål for, hvor stor en del af operatørens tid som bruges på reparation. Endelig er l et mål for det tab, som opstår ved at der står maskiner og venter på at komme til reparation. Det virker rimeligt at stræbe mod maximalt a_m og b_m, og minimalt l.

For at knytte disse størrelser til, hvad vi tidligere har lavet, må vi regne lidt på dem; vi har umiddelbart, at

$$a_m + b_m + c_m = m$$

da der altid for enhver maskine vil gælde, at den enten arbejder, er til reparation eller afventer reparation. Videre har vi

$$\frac{a_m}{b_m} = \frac{\mu}{\lambda} = \frac{1}{\rho},$$

da a_m/b_m er gennemsnitlig arbejdstid divideret med gennemsnitlig reparationstid.

For at udtrykke a_m noterer vi os, at taget over en meget lang tidshorisont (hvad man skal når man ser på gennemsnit) vil der være lige mange arbejdsperioder og reparationsperioder; vi kan derfor finde a_m ved at tage gennemsnitligt antal maskintimer i arbejds- og reparationsperioder, dvs. $v_m + v'_m$, og dividere med $x_m + x'_m$, som er gennemsnitlig længde af disse perioder tilsammen. Ialt får vi dermed

$$a_m = \frac{v_m + v'_m}{x_m + x'_m}.$$

Heri skal vi så indsætte fra det foregående, hvilket fører til udtrykket

$$a_m = \frac{1}{D}, \text{ hvor } D = \rho + \frac{1}{my_m}.$$

Vi får da

$$b_m = \rho a_m = \frac{\rho}{D},$$

og

$$c_m = m - (1+\rho)a_m = \frac{1}{D}[\frac{1}{y_m} + (m-1)\rho - 1]$$

Bruges nu f.eks. udtrykket for y_m fundet ovenfor, kan man i konkrete situationer regne sig frem til systemets karakteristika. Valget af m kan da foretages så karakteristika bliver bedst mulige under hensyn til omkostningerne.

7. Opgaver

1. En mindre finansforetagende har specialiseret sig i at forstrække mindre værtshuse med kredit til varelager. Hvis værtshusejeren misligholder sin gæld, overtager virksomheden værtshuset og sælger det til en ny ejer.

Værtshusene kommer ret tit ud for økonomisk uføre. Virksomheden har kendskab til, at ud af byens 127 værtshuse har 20 været konkursramt indenfor det sidste år, 28 i året før. 37 for to år siden, 10 for fire år siden, og 12 for fem år siden.

Virksomheden plejer at få 110% af sit tilgodehavende tilbage ved at overtage værtshuset og sælge det fordelagtigt.

Markedsrenten er 7%. Kan virksomhedens aktivitet betale sig?

2. Den amerikanske mangemilliardærdatter Nancy Plattbeater har i sit omtumlede liv været igennem adskillige ægteskaber af noget forskellig varighed:

Ægteskab nr.:	Ægtemand	Varighed (år)
1	Samuel Spratt (skuespiller)	1
2	Sir Robert Fitz-Martin (godsejer)	9
3	Bobby Frankenheim (bodybuilder)	1/2
4	James Green (playboy)	3
5	William Shorowitz (musiker)	12
6	Arthur Pennington (pensioneret oberst)	1
7	Ronald Rock (cowboy)	7
8	Johnny Sprumosa (fhv.gangster)	1

En ægteskabssvindler overvejer at tilbyde den nuværende ægtemand et betalt ophold i Las Vegas for i mellemtiden at overtale Nancy til at blive skilt og indgå nyt ægteskab. Der er givet tilbud fra et luksushotel (kost, logi, fri bar og kasino samt det løse) på 240.000 dollars pr. måned.

Hvor store omkostninger må ialt påregnes? Forklar forudsætningerne.

3. En lille virksomhed med 6 ansatte har lokaler på 5. sal i en ældre ejendom uden elevator. Virksomheden råder over et toilet, som befinder sig i ejendommens kælderetage. Hvis der er optaget, venter medarbejderne i nærheden.

Det har vist sig, at et gennemsnitligt toiletbesøg varer 10 minutter, og at medarbejderne føler behov for at gå på toilettet ca. en gang i timen. Nettoindtjeningen pr. medarbejder i arbejde er 374 kr. pr. time. Der arbejdes 8 timer om ugen i 50 uger pr. år. Markedsrenten er 10% p.a.

Der arbejdes med to forslag til forbedring af tilstanden. Ved det første indføres en ordning, hvor medarbejderne stempler ud ved toiletbesøg og stempler ind, når de vender tilbage. Der er forhandlet med fagforeningen, som accepterer ordningen mod et tillæg på 13 kr. i timen for hver medarbejder, hvortil kommer en engangsudgift på 35.000 kr. for ur og kort. Det andet forslag er at indrette toilet på 5.sal, hvad der koster 98.000 kr.

Skal virksomheden gennemføre nogle af disse forslag og i så fald hvilke? Diskutér forudsætninger.

8. Litteratur

En generel fremstilling af den sandsynlighedsteori, som ligger bag kapitlet, kan findes bl.a. i klassikeren Feller (1950, 1956). Fremstillingen af fornyelsesprocesser følger Borovkov (1986).

Maskinproblemet er egentlig et køproblem. Det er behandlet af adskillige forfattere; fremstillingen følger Goddard (1963).

KAPITEL 15

Pålidelighed og vedligehold

1. Binære pålidelighedssystemer

Teorien om pålidelighed (eng.: reliability) behandler statistiske egenskaber ved systemer bestående af mange enkelte komponenter. Hensigten med teorien er at kunne regne sig frem til brugbare vurderinger af varighed, sandsynlighed for nedbrud, værdien af vedligeholdsindsats osv. på grundlag af kendskab til de enkelte systemkomponenter. Tankegangen er den, at et system for at kunne fungere ikke nødvendigvis kræver, at alle dets komponenter er i orden; der er ofte en vis margin, således at dele af systemet kan være ude af drift uden at hele systemet er nede. Såfremt komponenternes svigt sker stokastisk, kan man fra komponenterne parametre finde systemets overordnede egenskaber.

Vi starter med det simpleste tilfælde, nemlig det, hvor de enkelte komponenter kan være i to tilstande, funktionsduelige og defekte; sådanne systemer kaldes *binære pålidelighedssystemer.* For hver komponent i (hvor $i = 1, \ldots, n$) indfører vi tilstandsvariablen x_i ved

$$x_i = \begin{cases} 1 & \text{hvis komponent } i \text{ fungerer,} \\ 0 & \text{hvis komponent } i \text{ er ude af funktion.} \end{cases}$$

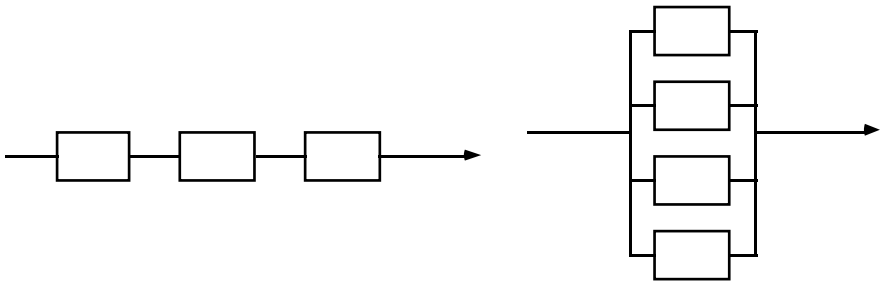
Vektoren af alle tilstandsvariable betegnes med x. Vi har tilsvarende for *hele systemet* en tilstandsvariabel z, der igen er binær; denne variabels værdi er bestemt af, hvorledes systemets enkelte komponenter har det; z er en funktion ϕ af tilstandsvektoren x. Vi har således

$$z = \phi(x) = \begin{cases} 1 & \text{hvis systemet fungerer,} \\ 0 & \text{hvis systemet er ude af funktion.} \end{cases}$$

I det binære tilfælde er der ikke andre muligheder. Funktionen $\phi : \{0,1\}^n \to \{0,1\}$ kaldes systemets *struktur funktion*.

Hvorledes strukturfunktionen ser ud, afhænger af, hvorledes systemet er skruet sammen. Hvis systemet er en serieforbindelse af komponenter (figur 1(a)), får vi

$$\phi(x) = \min_{1 \le i \le n} x_i;$$



(a) (b)

Figur 1

hvis systemet er en parallelforbindelse af komponenter (figur 1(b)), bliver den

$$\phi(x) = \max_{1 \leq i \leq n} x_i.$$

Fortolkningen af serie- og parallelforbindelse i det foregående er naturligvis bredere end blot at dække elektriske kredsløb; forbindelsen mellem komponenterne kan være af helt anden natur, f.eks. leverance af mellemprodukter i en produktionsproces. Generelt behøver der slet ikke at være nogen fysisk strøm eller økonomisk relation mellem komponenterne; det er strukturfunktionen selv, der er det grundlæggende begreb og som definerer systemet for os i det følgende.

Vi skal bruge et par nye begreber om strukturfunktioner: En komponent i er *irrelevant* for systemet ϕ hvis ϕ er konstant i x_i. Det sidste betyder, at det ikke har nogen betydning for strukturfunktionens værdi (systemets funktionsduelighed), om x_i er 0 eller 1, dvs. om komponenten virker eller ej. Hvis komponent i ikke er irrelevant, siger vi (naturligvis), at den er *relevant*.

Umiddelbart kunne man tro, at irrelevante komponenter er overflødige; det er imidlertid forkert. Det er kun komponentens øjeblikkelige tilstand, som ikke har betydning for systemets øjeblikkelige funktionsduelighed. Tilstedeværelsen af komponenten kan meget vel påvirke *sandsynligheden* for nedbrud af de andre komponenter. Tag f.eks. inderskærmene i en bil; man kan sagtens køre ud dem – det er der mange, der gør – men sandsynligheden for rustskader i karosseriet, og dermed på sigt kassering af hele bilen, nedsættes væsentligt af inderskærmene.

Hvis vi indfører notationen $x = (x_i, x_{)i(})$ for vektoren x (hvor vi har skrevet i'te komponent først, resten bagefter), kan vi definere komponent i's *strukturelle vægt* ved

$$I_\phi(i) = \frac{1}{2^{n-1}} \sum_{x:x_i=1} [\phi(1, x_{)i(}) - \phi(0, x_{)i(})].$$

Vi har simpelthen taget gennemsnittet over alle tilstande af systemet af forskellen i systemets funktion alt efter om komponent i er i drift eller ej. Hvis komponenten i er irrelevant, får vi igen, at $I_\phi(i) = 0$.

Strukturfunktionen ϕ siges at være *monoton,* hvis der for alle tilstande $x, y \in \{0,1\}^n$ gælder

$$x \geq y \Rightarrow \phi(x) \geq \phi(y).$$

At strukturfunktionen er monoton betyder, at systemet ikke bliver dårligere, hvis nogle af komponenterne bliver bedre. En strukturfunktion kaldes *koherent,* hvis den er monoton og ikke har irrelevante komponenter.

Vi kan omskrive strukturfunktionen på en praktisk måde ved at indføre begreberne en *snitmængde* og en *vej:* En snitmængde for en strukturfunktion ϕ er en delmængde C af komponenterne $\{1, \ldots, n\}$, således at hvis $x_i = 0$ for alle $i \in C$, $x_i = 1$ for alle $i \notin C$, da er $\phi(x) = 0$. En minimal snitmængde (der naturligvis er en snitmængde, som er minimal for inklusion) er således en mindste mængde C med den egenskab, at hvis de komponenter, der er defekte, netop er komponenterne i C, da vil systemet være ude af funktion.

En vej for strukturfunktionen ϕ er tilsvarende en mængde P af komponenter, som sikrer systemets funktionsduelighed, således at forstå at hvis $\{i|x_i = 1\} = P$, da er $\phi(x) = 1$.

Der gælder følgende:

Hvis ϕ er en monoton strukturfunktion med minimale snitmængder $C_1, \ldots, C_m$ og minimale veje $P_1, \ldots, P_r$, da er

$$\phi(x) = \min_{1 \leq j \leq m} \max_{i \in C_j} x_i = \max_{1 \leq j \leq r} \min_{i \in P_j} x_i.$$

Som begreberne er defineret, vil vi have $\phi(x) = 0$ hvis og kun hvis alle komponenter i en minimal snitmængde C_j er 0, hvilket kræver at

$$\max_{i \in C_j} x_i = 0.$$

På den anden side vil vil have, at ingen af de minimale veje kan bestå af funktionerende elementer, hvilket helt tilsvarende betyder, at $\min_{i \in P_j} x_i = 0$ for hvert j. Vi har dermed, at udtrykket er rigtigt for alle x med $\phi(x) = 0$. Tilfældet $\phi(x) = 1$ klares på helt tilsvarende måde. □

2. Pålidelighedsmål

Indtil nu har der ikke være stokastik inddraget. Vi vil nu antage, at komponenternes tilstand følger hver deres stokastiske proces $X_i(t)$. Den samlede vektor af komponenternes tilstande bliver altså en n-vektor af 0 eller 1, og den følger en vektor stokastisk proces $(X_1(t), \ldots, X_n(t))$; videre har vi en stokastisk proces $Z(t)$ med værdier i $\{0,1\}$ givet ved

$$Z(t) = \phi(X_1(t), \ldots, X_n(t)).$$

Med denne formulering kan vi diskutere en række oplagte mål for systemets overlevelsesevne:

Systemets *første passagetid* er den stokastiske variabel

$$T = \inf\{t|Z(t) = 0, t \geq 0\},$$

som er den tid, der går, inden systemet bryder ned første gang. *Pålideligheden* af systemet, givet ved

$$R(t) = P\{T > t\} \text{ for } t \geq 0$$

er sandsynligheden for at systemet ikke er brudt ned en eneste gang fra start og frem til tidspunkt t. Pålideligheden af et system er et centralt begreb, og det er naturligvis væsentligt at kunne regne sig frem til denne så præcist som muligt. Det vender vi tilbage til om lidt.

Hvis systemet betragtes i tidspunkt t, har vi tilsvarende tiden til næste nedbrud defineret ved

$$T_t = \inf\{s : Z(t+s) = 0, s \geq 0\},$$

der igen er en stokastisk variabel. Vi har $T_t = 0$ hvis systemet er nede i tidspunkt t. Pålideligheden i tidspunkt t er givet ved

$$R_t(s) = P\{T_t > s\} \text{ for } s \geq 0.$$

Hvis $R_t(s)$ går mod en grænse $R_\infty(s)$ for $t \to \infty$ kaldes denne den asymptotiske pålidelighed af systemet.

Hvis systemet har været i funktion op til og med tidspunkt t, kan vi finde den betingede sandsynlighed for, at det fungerer efter yderligere s som

$$P\{Z(s) = 1, t \leq \tau \leq s|Z(\tau) = 1, 0 \leq \tau \leq t\} = \frac{R(t+s)}{R(t)}.$$

Grænsen af dette forhold for t gående mod uendelig, dvs.

$$R_Q(s) = \lim_{t\to\infty} \frac{R(t+s)}{R(t)}$$

(der her forudsættes veldefineret) kaldes den *quasi-stationære pålidelighed* og giver os en tilnærmelse til systemets pålidelighed efter at det har arbejdet i lang tid.

Det betragtede system kan enten være kassabelt når det bliver funktionsudueligt (f.eks. en satellit), eller det kan repareres. Begreberne ovenfor giver lige fuld mening i begge tilfælde; de, der følger nedenfor, relaterer til systemer, der kan komme i funktion igen efter nedbrud. Sådanne systemer er henholdsvis "nede" og "oppe", og bevægelsen fra det ene til det andet foregår stokastisk. Antag, at systemet er oppe i tidspunkt 0; vi sætter $R_0 = 0$ og definerer i rækkefølge størrelserne

$$F_k = \inf\{t|Z(t) = 0, t > R_{k-1}\},\ k = 1, 2, \ldots,$$

som er tidspunktet for k'te nedbrud (en stokastisk variabel), og

$$R_k = \inf\{t|Z(t) = 1, t > f_k\},\ k = 1, 2, \ldots,$$

der angiver det stokastiske tidspunkt, hvor systemet er repareret efter k'te nedbrud (disse stokastiske variable kan eventuelt tage værdien $+\infty$). Vi har så

$$U_k = F_k - R_{k-1},\ k = 1, 2, \ldots,$$

$$D_k = R_k - F_k,\ k = 1, 2, \ldots,$$

for varighederne af de forskellige "oppe"- og "nede"-perioder. En enkelt af dem kendte vi allerede, nemlig $T = U_1$. Fra disse stokastiske variable kan vi få nogle mere håndterlige størrelser til mål af systemets funktion, f.eks. gennemsnitlig funktionsperiode (også kaldet mean time to failure)

$$\text{MTTF} = \lim_{k\to\infty} E(U_k)$$

og gennemsnitlig reparationsperiode (mean time to repair)

$$\text{MTTR} = \lim_{k\to\infty} E(D_k).$$

Antag at systemet har været i funktion længe (med en blandet forhistorie af "oppe" og "nede") og at det netop er repareret. Det er da af interesse at kende pålideligheden netop her. Vi har et må for dette i den såkaldte post recovery pålidelighed

$$R_P(s) = \lim_{k\to\infty} P\{U_k > s\}.$$

Punktdueligheden (point availability) $A(t)$ er sandsynligheden for, at systemet virker på tidspunkt t,

$$P\{Z(t) = 1\};$$

den er lig med $R(t)$ for et system, som ikke kan repareres. *Intervaldueligheden* $\text{AI}(t, \tau)$ er den gennemsnitlige del af et interval af længden τ startende i t, i hvilken systemet fungerer; dette kan også skrives

$$\text{AI}(t, \tau) = \frac{1}{\tau} \int_t^{t+\tau} A(s)ds.$$

Hvis grænsen $A = \lim_{t\to\infty} A(t)$ eksisterer, da er

$$A = \lim_{\tau\to\infty} \text{AI}(t, \tau) = \lim_{t\to\infty} A(t, \tau).$$

3. Pålidelighed i specielle situationer

Vi har nu indført de gængse mål for et systems holdbarhed. Spørgsmålet er så, om vi kan regne os frem til nogle eller samtlige af disse i konkrete situationer.

Systemer, som ikke kan repareres. Vi ser først på tilfældet, hvor systemet *ikke* kan repareres. Vi skal videre antage, at systemets komponenter i har stokastiske levetider T_i, som er stokastisk uafhængige.

Før vi indfører stokastik, noterer vi, at der gælder vi følgende nye udgave af sætningen fra afsnit 1:

Hvis $C_1, \ldots, C_m$ og $P_1, \ldots, P_r$ er de minimale snitmængder og minimale veje for strukturfunktionen ϕ, da gælder

$$T = \min_{1 \le j \le m} \max_{i \in C_j} T_i = \max_{i \le j \le r} \min_{i \in P_j} T_i.$$

For at bevise dette behøver vi blot notere os, at den tid, der går, før alle komponenter i snitmængden C_j er havareret, er $\max_{i \in C_j} T_i$. Siden systemet bryder sammen når blot en snitmængde er nede, har vi det første lighedstegn. Det andet følger ved en tilsvarende overvejelse. □

Antager vi nu, at T_i har sandsynlighedsfordelingen F_i (således at $F_i(t)$ er sandsynligheden for, at komponent i er gået i stykker inden t), og alle T_i er stokastisk uafhængige, kan vi udregne *pålideligheden:* Vi indfører de stokastiske processer $X_i(t)$ givet ved

$$X_i(t) = \begin{cases} 1 & \text{hvis } T_i > t \\ 0 & \text{hvis } T_i \le t \end{cases}$$

og har da

$$\begin{aligned} R(t) &= \mathrm{E}\phi(\mathrm{X}_1(\mathrm{t}), \ldots, \mathrm{X}_\mathrm{n}(\mathrm{t})) \\ &= \sum_{x \in \{0,1\}} \phi(x) \prod_{i=1}^{n} (1 - F_i(t))^{x_i} F_i(t)^{1-x_i}; \end{aligned}$$

Udtrykket ser måske indviklet ud, men argumentet bag det er ligetil: Når vi skal finde middelværdien af udtrykket i første linie, skal vi for hver værdi af $x = (x_1, \ldots, x_n)$ (som er en vektor af 0 og 1) finde sandsynligheden for netop dette x. Denne sandsynlighed er produktet af de enkelte koordinaters sandsynligheder (på grund af den stokastiske uafhængighed), og sandsynligheden for $x_i = 1$ på tidspunkt t er $1 - F_i(t)$; for $x_i = 0$ på tidspunkt t er den $F_i(t)$. Det er faktisk bare det, der står i formlen; den ene af de to faktorer er altid 1, idet potenserne altid er enten 0 eller 1.

Iøvrigt kommer der ikke så mange spændene pålidelighedsmål ud af vore overvejelser i denne situation, hvor systemet ikke kan repareres. Punktdueligheden

er som nævnt lig med $R(t)$, og intervalduelighedenfindes som

$$\mathrm{AI}(t,\tau) = \frac{1}{\tau}\int_t^{t+\tau} R(s)ds.$$

Hvis $F_i(s) \to 1$ for $t \to \infty$, er alle de asymptotiske mål 0.

Systemer, der kan repareres. Lad os derefter antage, at komponenterne kan repareres, således at systemet har oppe- og nedeperioder U_k og D_k og starter med at være "oppe" på tidspunkt 0. Vi antager at følgerne (U_k) og (D_k) er uafhængige stokastiske variable med fordelingsfunktioner F_U og F_D. Vi kan da finde et udtryk for punktdueligheden på tidspunkt t. Hertil skal vi bruge lidt notation:

Lad de to stokastiske variable X_1, X_2 have fordelingsfunktioner F_1 og F_2 og tilhørende tæthedsfunktioner f_1 og f_2. *Foldningen* $F_1 \star F_2$ af F_1 og F_2 defineres som fordelingsfunktionen med tæthed

$$\int_{-\infty}^{\infty} f_1(t-s)f_2(s)ds.$$

Dette er naturligvis fordelingsfunktionen for summen af X_1 og X_2. Hvis vi tager foldningen af fordelingsfunktionen F med sig selv k gange, skriver vi $F^{(k)} = F\star\cdots\star F$. Med denne notation kan vi skrive punktdueligheden i tidspunkt t som

$$A(t) = (1-F_U(t)) + \sum_{k=1}^{\infty}\int_0^t (1-F_U(t-\tau))d(F_U^{(k)} \star F_D^{(k)}(\tau))$$

(hvor vi har skrevet fordelingsfunktionen efter d'et under integraltegnet; det kan man her blot opfatte som en konvention, således at vi lige så godt kunne have skrevet den tilhørende tæthedsfunktion foran d'et, efterfulgt af $d\tau$. Der er en vis pointe i denne skrivemåde, idet integralet også giver mening for fordelinger, der ikke har nogen tæthedsfunktion, men det kan vi tage afslappet her).

I udtrykket står der, at sandsynligheden for, at systemet virker på tidspunkt t, er summen af sandsynlighederne for at systemets første nedbrud sker efter t (det er første led i udtrykket) og sandsynlighederne for, at der har været k nedbrud med efterfølgende afsluttet reparation inden t, for k et vilkårligt tal større end 1. Udtrykket kan skrives lidt kortere, hvis vi noterer os, at

$$\sum_{k=1}^{\infty} F_U^{(k)} \star F_D^{(k)}(\tau) =$$

$$\sum_{k=1}^{\infty} kP\{\text{der er gennemført } k \text{ reparationer indtil } \tau\}$$

altså det gennemsnitlige antal reparationer i intervallet $[0, \tau]$, som vi vil betegne med $M_R(\tau)$. Derved fås udtrykket

$$A(t) = (1 - F_U(t)) + \int_0^t (1 - F_U(t - \tau))dM_R(\tau),$$

der ihvertfald formelt er noget simplere, men selv i denne udgave vil det som regel være vanskeligt at bruge i praksis. Man vil som oftest være nødt til at nøjes med grænseværdien A.

Denne grænseværdi kan vi finde ved hjælp af Blackwell's sætning: Vi har de to uafhængige fornyelsesprocesser (U_k) og (D_k), og vi ved da, at det gennemsnitlige antal nedbrud pr. tidsenhed er $1/\mathrm{EU}$; det svarer til, at den gennemsnitlige varighed af en "oppe"-periode er EU. Tilsvarende er den gennemsnitlige varighed af en reparation ED. Ialt har vi dermed ved en simpel gennemsnitsbetragtning, at den andel af en tidsenhed, i hvilket systemet virker, må være

$$A = \frac{\mathrm{EU}}{\mathrm{EU} + \mathrm{ED}}.$$

Lad os prøve at gennemføre et ræsonnement af tilsvarende art for pålideligheden. Vi har for det første, at

$$R_t(\tau) = 1 - F_U(t + \tau) + \int_0^\infty (1 - F_U(t + \tau - s)dM_R(s);$$

igen er vi til praktiske formål mere interesserede i den asymptotiske værdi. Vi skal have fat i sandsynligheden for, at systemet fungerer i en periode af længde s regnet fra t, hvor t er meget stor. Det er ensbetydende med at systemet fungerer på tidspunkt t *og* bliver ved med det i s tidsenheder. Sandsynligheden for den første begivenhed er A fundet ovenfor; den anden begivenheds sandsynlighed finder vi som

$$\frac{\int_s^\infty (1 - F_U(t)dt)}{\mathrm{EU}}.$$

Kombineres disse, fås

$$R_\infty(\tau) = \frac{\mathrm{EU}}{\mathrm{EU} + \mathrm{ED}} \frac{\int_\tau^\infty (1 - F_U(t)dt}{\mathrm{EU}}.$$

Multipel komponent systemer. Vi kan kombinere de to foregående afsnit, således at vi ser på systemer, der består af mange komponenter, der hver især kan repareres. Det er først når tilstrækkelig mange af komponenterne er defekte, at hele systemet bryder sammen (og ikke længere kan repareres). Komponenternes processer antages uafhængige.

Vi kan nu bruge forrige afsnits resultater til at udregne dueligheden $A_i(t)$ for hver komponent. Dernæst sættes denne ind i pålidelighedsfunktinen h for hele systemet. specielt får vi dermed

$$A = h(\frac{\mu_1}{\mu_1 + \lambda_1}, \ldots, \frac{\mu_n}{\mu_n + \lambda_n})$$

hvor μ_i og λ_i er den gennemsnitlige "oppe"- og "nede"-tid for komponent i.

Pålideligheden $R(t)$ lader sig desværre ikke udtrykke helt så simpelt som A; i de fleste praktisk situationer må man nøjes med at finde øvre og nedre grænser for dens værdi.

4. Opgaver

1. En alfons har monopol i en mindre provinsby. Han råder over 5 piger, der trækker på hver sin af byens hovedgader. Der er en vis risiko for, at en pige bliver anholdt af byens sædelighedspoliti, hvorved hun idømmes 2 måneders fængsel. Den særlige afdeling af byretten, der tager sig af løsgængeri, har en ret langsom sagsbehandling, og en anholdt pige bliver derfor varetægtsfængslet efter razziaen indtil sagen kan komme for retten. I gennemsnit er der razzia på hovedgaderne hver tredie måned.

Når en pige er i arbejde, indtjener hun 35.000 kr. pr. måned til alfonsen. Skattemyndighederne har på dette grundlag sat indtægten til 175.000 om måneden. Hertil svarer hans advokat, at der må kunne fratrækkes for driftstab. Hvad er den korrekte gennemsnitlige indtægt?

5. Litteratur

Problemstillingen om systempålidelighed dukkede op i efterkrigsårene, blandt andet i forbindelse med overvejelser om kommunikation og computerteknik. Senere er det f.eks. miljøproblemer, som har givet anledning til overvejelser om pålidelighed. En bred fremstilling kan findes i Ascher og Feingold (1984).

KAPITEL 16

Det optimale lager

1. Indledning

Lageret (eller rettere lagrene, for der er som regel flere) er en vigtig bestanddel af enhver virksomhed. Det, at der findes et lager af varer til salg, eller halvvejs igennem produktionen, eller måske af råvarer, sikrer, at produktions og afsætning kan forløbe uhindret; man bliver ikke nødt til at sende kunder bort fordi varen ikke haves, eller at stoppe produktionen fordi man ikke har stumperne.

Så vidt er det hele ganske ligetil, men vi er selvfølgelig ikke færdige ved blot at observere, at det er vigtigt at have et lager. Der resterer naturligvis problemet om at finde det rigtige – eller, i vor sædvanlige sprogbrug – det *optimale lager*. Det er det, vi skal beskæftige os med her.

For at finde det optimale lager må vi naturligvis tage rede på, hvilke fordele og ulemper der er forbundet med lagre af den ene eller den anden type, og vi må se nærmere på, hvad der faktisk kan kontrolleres i forbindelse med lagerholdet. Hvad det sidste angår, er vi især interesseret i at fastlægge *hvor meget der købes til lager*, og *hvor ofte*. Fordele og ulemper ved lagerhold af forskellig art skal altså ses i relation til disse to typer beslutningsvariable. Sådan set burde vi også inddrage overvejelser om *hvad* der lægges på lager (hvor færdige er varerne?), men det vil vi ikke komme ind på her.

Ved undersøgelsen af de økonomiske konsekvenser af forskellige politikker mht. lageret er det almindeligt at opgøre alt som *omkostninger*; der foreligger følgende forskellige typer:

- Lagerhold: Omkostninger som er proportionale med lagerets størrelse, f.eks. rente af den kapital der er investeret i lageret (bemærk at denne slags omkostning normalt ikke opgøres særskilt i regnskabet; hvis der ikke er lånt kapital i virksomheden, vil den slet ikke kunne ses. Ikke desto mindre kan de med nogen ret ses som en omkostning på lige fod med sådanne der faktisk betales; det er en alternativomkostning; man kunne have fået rente af den kapital der sidder i lageret). Videre indgår her lønomkostning til lagerarbejdere, leje af lokaler, forsikring osv.

- Knaphedsomkostninger, der indtræder hvis en vare, som ønskes, ikke findes på lager, f.eks. de ekstraomkostninger der går til at fremskaffe den (telex, luftfragt osv.),

- Omkostninger ved skift i produktionsrytme, som løber på hvis lageret ikke er stort nok og må suppleres hurtigt,
- Tab af rabatter ved hasteindkøb.

Disse forhold skal naturligvis vurderes i forhold til behovet for lager, repræsenteret ved efterspørgslen efter varen (som ikke nødvendigvis er kundernes efterspørgsel – det vil det være for et færdigvare lager, men ikke for et lager brugt internt), samt leverancerne til lageret. Disse to sidste variable, der repræsenterer henholdsvis af- og tilgang til lageret, kan være formuleret deterministisk eller stokastisk. Vi starter med det deterministiske tilfælde.

2. Det deterministiske lagerproblem

I det følgende ser vi på det simpleste lagerproblem, nemlig fastsættelse af den optimale lagerpolitik i en situation, hvor efterspørgslen efter varen er helt konstant over tiden, således at et givet lager afvikles med konstant hastighed λ (enheder pr. tidsenhed). Da problemstillingen er kendt fra driftsøkonomien, skal vi gøre det kort.

Vi antager et regelmæssigt tilbagevendende forløb, hvor der startes med et lager af maximal størrelse $Q + s$; i løbet af T tidsenheder bliver dette lager reduceret til s, hvorfor der må gælde

$$\lambda T = Q.$$

Vi vil også operere med et ordre-lag af størrelsen τ tidsenheder; hvis lageret skal have tilført nye varer på et vist tidspunkt t skal ordren herom altså afgives ved tidspunkt $t - \tau$. Det lager, der forefindes, når ordren afgives, betegnes med s. Bemærk, at det foreløbig *ikke* tager tid at fylde lageret; τ er her blot en forsinkelse fra beslutningen tages til den effektueres.

I denne indledende betragtning er det således Q, det mængde, der fyldes på lageret, og s, den lagerstørrelse, ved hvilken der opfyldningen sker, som er beslutningsvariable. Strengt taget har vi også tidspunktet for ordreafgivelse, men dette tidspunkt må nødvendigvis være $T - \tau$, og det lager, ved hvilken der afgives ordre om ny opfyldning, er altså

$$(T - \tau)\lambda.$$

For at finde Q og s skal vi inddrage omkostningerne ved lagerpolitikken. Vi antager (som vanlig), at der er to slags, nemlig *lagerholdsomkostninger*, der antages proportionale med aktuel lagerstørrelse, dvs.

$$\text{lagerholdsomk.} = \int_0^T a(Q + s - \lambda t)dt = aT\left(\frac{Q}{2} + s\right),$$

og *ordreomkostninger,* som er et vist beløb b ved hver opfyldning uanset dennes størrelse.

De samlede omkostninger ved en fast periode af varighed 1 tidsenhed (i løbet af hvilken der sker ialt $1/T$ gennemløb af lagercyklus'en) bliver således

$$C = \frac{a}{T}T\left(\frac{Q}{2} + s\right) + \frac{b}{T},$$

som, når vi indsætter $T = Q/\lambda$, bliver til

$$C = a\left(\frac{Q}{2} + s\right) + \frac{\lambda}{Q}b.$$

Vi finder den optimale politik ved at minimere dette udtryk. Nødvendige første ordens betingelser er

$$\frac{\partial C}{\partial Q} = \frac{a}{2} - \frac{\lambda b}{Q^2} = 0$$

med hensyn til Q, som ved løsning giver den velkendte *kvadratrodsformel*

$$Q^* = \sqrt{\frac{2\lambda b}{a}}.$$

Det er åbenbart, at s skal vælges til 0, da den indgår med positiv vægt i udtrykket for C (vi tager ikke partiel afledet og sætter til 0, for minimum antages tydeligt nok på randen).

Hvis den underliggende problemstilling er mere kompleks end dette tilfælde, vil det resulterende udtryk også blive en smule mere kompliceret. Vi ser først på den situation, hvor *der kan være negativt lager.*

Negativt lager, eller *restordre,* som det kaldes, er ikke så mærkeligt et fænomen som det umiddelbart kunne se ud til. Det er ikke usædvanligt, at kunder i tilfælde af tomt lager accepterer at vente på at blive forsynet, idet de naturligvis får deres varer før der lægges på lager igen. Den ulempe, som den manglende eller forsinkede levering forvolder kunderne, kan tænkes godtgjort ved en passende betaling; denne betaling, der antages at være en godtgørelse a' pr.stk. og pr.tidsenhed, er lageromkostningerne ved negativt lager, svarende til de omkostninger, man har ved at opretholde et positivt lager.

Når det tager tid at fylde lageret op, idet dette tænkes at ske med en hastighed ψ stk. pr. tidsenhed, bliver der forskel på ordren Q til lageret og det, der faktisk kommer på lager. I opbygningsperioden kommer der varer ind på lageret med hastigheden $\psi - \lambda$. Produceres der i perioden T_p, har vi $T_p = Q/\psi$, og den samlede lageropbygning bliver

$$Q' = T_p(\psi - \lambda) = Q\left(1 - \frac{\lambda}{\psi}\right).$$

Den samlede lageropbygning nedbringes da til det minimale lager s (der nu godt kan være negativt) i løbet af en periode T_d, der kan findes fra udtrykket

$$T_d\lambda = Q',$$

dvs. som

$$T_d = \frac{Q}{\lambda}\left(1 - \frac{\lambda}{\psi}\right).$$

Vi skal nu tilføje problemet med negativt lager: Ud af perioden T_d vil en vis tid forløbe med negativt lager, nemlig T_{dn} givet ved

$$\max\{0, -s\}\lambda = T_{dn}$$

(det lidt indviklede udtryk $\max\{0, -s\}$ sikrer, at T_{dn} er 0 når der i politikken er vedtaget et $s \geq 0$, så at der slet ikke optræder negativt lager. Tilsvarende vil der være en del af opfyldningsperioden, nemlig T_{pn} givet ved

$$\max\{0, -s\}(\psi - \lambda) = T_{pn},$$

hvor der er negativt lager.

For en enkelt lagercyklus har vi nu dels omkostningerne ved opfyldning b (der nu ofte kaldes produktionsforberedelsesomkostninger), dels lagerholdsomkostninger. Disse kan igen deles i to, nemlig omkostningerne i perioden med negativt lager, ialt

$$(T_{dn} + T_{pn})\frac{(-s)}{2}a' = \max\{0, -s\}\psi\frac{(-s)}{2}a',$$

og omkostningerne i perioden med positivt lager,

$$((T_p - T_{pn}) + (T_d - T_{dn}))\frac{(Q' - \max\{0, -s\})}{2}a.$$

Ved indsættelse af udtrykkene for T_p, T_{pn}, T_d, T_{dn} og Q' fås et udtryk, som kun indeholder variablene Q og s samt parametrene. Ved minimering af de samlede omkostninger i en periode af varighed 1 kan Q og s findes. Da de konkrete formler ikke er særlige instruktive, vil vi ikke gå ind på dem her.

Disse udtryk for det optimale lager – under de simplificerende betingelser diskuteret ovenfor, nemlig

- efterspørgsel af fast, kendt størrelse,
- ingen forsinkelse mellem ordre til lager og levering til lager,
- øjeblikkelig produktion
- negativt lager ikke tilladt

er klassiske og har fundet udbredt anvendelse, ikke alene i praktisk lagerstyring, men også i andre forbindelser Den har naturligvis de begrænsninger, som er indeholdt i forudsætningerne ovenfor. Blandt disse er det især antagelsen om kendt – og deterministisk – afsætning fra lageret, som er problematisk. Men også på andre punkter kunne man ønske større realisme.

Variable produktionsomkostninger. I det foregående har vi antaget, at der udover éngangsomkostningen b ved at sætte produktion i gang (eller ved at købe ind til lageret) ikke var omkostninger ved lageropfyldningen. Det virker måske lidt tvivlsomt, men her skal det bemærkes, at det kun er, hvis produktionsomkostningerne afhænger ikke-lineært af seriestørrelsen Q, at der kommer noget nyt ind i vore overvejelser; hvis stykomkostningerne er konstante (når der bortses fra den faste omkostning), får de ikke nogen indflydelse på det optimale lagers størrelse.

Det er imidlertid ikke usædvanligt, at der ved lageropfyldning kan være indkøbsbetingelser, som resulterer i ikke-lineære omkostninger, f.eks. *kvantumsrabatter*. Antag f.eks., at der ved indkøb til lager gælder en pris p pr. stk. hvis der købes op til q' enheder, hvorefter prisen p' hvis der købes q' eller flere enheder. Vi får da en omkostningsfunktion

$$a(Q) = \begin{cases} 0 & Q = 0 \\ b + pQ & O < Q < q'. \\ b + q'p + (Q - q')p & q' \leq Q \end{cases}$$

Vi skal ikke forfølge dette problem yderligere; der er den tekniske vanskelighed, at omkostningsfunktion af denne type har et knæk i q', og det kræver lidt behændighed at finde optimum (det går ikke uden videre at differentiere og sætte lig nul, når de funktioner, man har med at gøre, ikke er differentiable). I det konkrete tilfælde er det nu ikke så svært at tilpasse metoderne; man kan beregne det optimale lagerhold ved hver af de konstante variable stykomkostninger p og p' og så efterfølgende sammenligne løsningerne.

3. Lagerpolitik ved stokastisk efterspørgsel: Aviskioskmodellen

Vi vil nu forbedre modellen fra det foregående ved at antage, at efterspørgslen ikke sker med en helt konstant størrelse pr. tidsenhed, men at den er stokastisk. Hvis vi i denne situation skal gennemføre overvejelserne fra forrige afsnit, således at vi finder en optimal lagerpolitik over tid, må vi formulere efterspørgslen som en stokastisk proces. Vi vil her i starten tage et noget simplere tilfælde, idet vi ser på det ganske særligt lagerproblem forbundet med salg af en vare, hvor der ikke kan genanskaffes til lager.

En simpel basismodel for lager ved stokastisk efterspørgsel udgøres er den såkaldte "aviskiosk"-model: Der er kun én periode, hvor der finder salg sted, men trækket på lageret er ikke kendt, når der bestilles til lager. Vi betegner dette

træk på lageret, efterspørgslen, med D, og vi antager, at D har en kontinuert fordelingsfunktion Φ med middelværdi μ. Der er ordreomkostninger c pr. enhed, og der er en omkostning på c_H for lagerhold, som beregnes pr. enhed lager ved periodens slutning; hvis der omvendt er restordre, beregnes c_P pr. enhed. Den stokastiske efterspørgsel antages ikke-negativ, så $\Phi(x) = 0$ for $x < 0$.

Vor beslutningsvariabel i modellen er ikke anskaffelse til lager, men *lagerstørrelse* efter ordreafgivelse (idet vi i denne enperiodemodel antager, at ordren leveres umiddelbart). Lad y være denne lagerstørrelse. Vi har da forventet omkostning ved lagerhold, respektive restordre, givet ved

$$L(y) = \int_0^y c_H(y-\xi)d\Phi(\xi) + \int_y^\infty c_P(\xi - y)d\Phi(\xi),$$

idet der skal tages gennemsnit over alle tænkelige værdier af efterspørgslen, her skrevet som ξ, der kan være såvel mindre end lageret (første led) som større end lageret. Den samlede omkostning, som er summen af ordreomkostninger og de øvrige omkostninger, bliver

$$\begin{aligned} g(y) &= cy + L(y) \\ &= c\mu + (c + c_H)\int_0^y (y-\xi)d\Phi(\xi) \\ &\qquad + (c_P - c)\int_0^\infty (y-\xi)d\Phi(\xi) \\ &= c\mu + (c+c_H)(y-\mu) + (c_P + c_H)\int_y^\infty (\xi - y)d\Phi(\xi). \end{aligned}$$

Vi skal nu variere på y således at $g(y)$ bliver minimal. Løsningen findes ved at differentiere g om sætte lig nul (da g er en konveks funktion, bliver det minimum, vi får frem):

$$g'(y) = c + c_H - (c_P + c_H)\int_y^\infty d\Phi(\xi) = 0$$

(idet vi har brugt kontinuiteten af Φ, så at vi kan differentiere integralet m.h.t. y). Vi får heraf

$$\int_y^\infty d\Phi(\xi) = \frac{c + c_H}{c_P + c_H}$$

eller, hvis vi kalder den optimale lagerstørrelse for S,

$$\Phi(S) = \int_0^S d\Phi(\xi) = 1 - \frac{c + c_H}{c_P + c_H} = \frac{c_P - c}{c_P + c_H}.$$

Vi har således, at S kan findes som fraktilen i Φ-fordelingen svarende til $(c_P - c)/(c_P + c_H)$, der kaldes *den kritiske fraktil*. Vi kan finde det optimale lagerniveau

S, når blot den kritiske fraktil faktisk er en sandsynlighed, dvs. når $c_P > c$ og $c_P + c_H > c_P - c$.

Betragtes udtrykket for $g(y)$, har vi et første led $c\mu$, der kan fortolkes som forventet omkostning *ved* perfekt information – hvorved der forstås, at man kan vente med at afgive lagerordre indtil man kender efterspørgslen – og de øvrige to led angiver forventet værdi *af* perfekt information, idet disse omkostningskomponenter jo netop ville forsvinde ved perfekt information. Denne størrelse kaldes ofte for *buffer omkostningerne,* idet de repræsenterer udgifterne ved at holde et buffer- eller stødpudelager, hvis formål ene er at opsamle uforudset variation i efterspørgslen.

Størrelsen $S - \mu$ angiver det optimale bufferlager. Afhængig af den kritiske fraktils værdi kan dette bufferlager være stort eller lille; det kan endda være negativt, svarende til at S er mindre end μ. Udtrykket for $\Phi(S)$ kan iøvrigt omskrives til

$$c_P(1 - \Phi(S)) = c + c_H\Phi(S),$$

som fortæller os, at marginal gevinst er lig med marginal omkostning: Hvis der lægges en enhed mere på lager, sparer vi restordreomkostningerne gange sandsynligheden for restordre (efterspørgsel større end S), mens vi betaler ordreomkostninger plus lageromkostning gange sandsynlighed for lager.

En oplagt specifikation af Φ gælder for tilfældet, hvor efterspørgslen stammer fra mange små kunder (hvis individuelle efterspørgsler er stokastisk uafhængige). Vi kan da antage, at D er normalfordelt; strengt taget har vi udelukket normalfordelingen, da vi antog $\Phi(\xi) = 0$ for $\xi < 0$, men vi kan tænke os normalfordelingen fremkommet som tilnærmelse (ved brug af den centrale grænseværdisætning), og så er værdierne på den negative akse alligevel så små, at vi kan se bort fra dem.

Lad σ være standardafvigelsen i den specificerede fordeling. Hvis vi skriver fordelingsfunktionen for $N(0,1)$ (normalfordelingen med middelværdi 0 og varians 1) som Φ_N og lader $z = \Phi_N^{-1}(\xi)$, har vi optimalt lager givet ved $S = \mu + z\sigma$, således at det optimale bufferlager er $z\sigma$. Den tilhørende optimale værdi af kriteriefunktionen kan skrives

$$c\mu + [(c + c_H)z + (c_P + c_H)I_N(z)]\sigma,$$

hvor $I_N(\cdot)$ er den normale tabsfunktion, som findes i tabelværker. Vi har i dette tilfælde, at såvel bufferlager som bufferomkostninger er proportionale med standardafvigelsen i efterspørgslen.

4. Stokastisk efterspørgsel: Flerperiode-modeller

Aviskiosk-modellen har været genstand for betydelig forskning, og der findes en omfattende litteratur om denne, idet en række af antagelser kan modificeres, mens grundprincipperne i analysen forbliver den samme.

Den første oplagte modifikation går ud på at indføre et *initialt lager* x. Kriteriefunktionen fra det foregående bliver nu

$$G(y;x) = c(y-x) + L(y) = g(y) - cx.$$

Det kan vises, at hvis blot g er quasi-konvex (hvilket dækker en ganske bred klasse af omkostningsfunktioner), så går den optimale lagerpolitik ud på at indkøbe nok til at bringe lagerets størrelse op på en bestemt størrelse S (således som det iøvrigt også var tilfældet i forrige afsnit), hvis $x \leq S$, og at gøre ingenting ellers. Denne politik kaldes for en *basislagerpolitik* eller en politik med *enkelt kritisk værdi*.

Dette kan bruges til at give et fingerpeg om, hvorledes den optimale lagerpolitik ser ud, når modellen udvides til at gælde for n perioder i stedet for blot en enkelt. Antag, at vi starter disse n perioder med et lager på x enheder. Lad $C_n(x)$ være middelværdi af tilbagediskonterede omkostninger ved en optimal n-perioders lagerpolitik. Vi har da umiddelbart, at $C_n(\cdot)$ opfylder ligningen

$$C_n(x) = \min_{y \geq x} \left\{ c(y-x) + L(y) + (1+r)^{-1} \int_0^\infty C_{n-1}(y-\xi) d\xi \right\}.$$

Herved får vi – ihvertfald formelt – flerperiode problemet reduceret til et problem af den allerede kendte type med en enkelt periode og initialt lager, hvor den optimale politik er typen med enkelt kritisk værdi. Uheldigvis har vi jo $C_{n-1}(\cdot)$ til at indgå, så problemet er ikke løst uden videre.

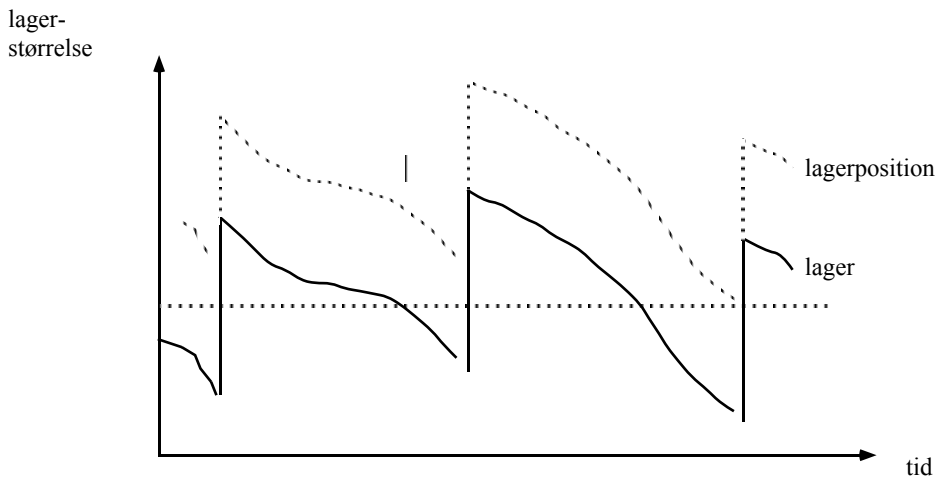
Det kan imidlertid vises, at den optimale politik over de n perioder har en ret simpel form (ihvertfald når $L(y)$ er konveks); den er karakteriseret ved en følge $(S_n, s_n), \ldots, (S_1, s_1)$; ved starten af n-periode problemet skal man supplere lageret op til S_n hvis $x \leq s_n$ og gøre ingenting ellers. Derefter går man over til et $n-1$-perioders lagerproblem.

En sådan politik med to kritiske værdier S og s, nemlig et kaldes normalt for en (S,s)-lagerpolitik. De konkrete værdier af S og s afhænger naturligvis af såvel omkostningsstruktur som efterspørgslens fordeling.

5. Lagerpolitik ved stokastisk efterspørgsel i kontinuert tid

Vi vender nu tilbage til teorien fra afsnit 2, idet efterspørgslen nu er formuleret som en stokastisk proces. Vi vil om denne proces blot antage, at efterspørgslen i hvert lille tidsinterval er uafhængig af efterspørgslen i andre tidsintervaller, og at middelværdien af efterspørgslen pr. tidsenhed er en konstant λ.

Når efterspørgslen ikke længere kan forudsiges med absolut sikkerhed, er der visse af detaljerne i lagerpolitikken fra de tidligere afsnit, der kommer til at spille en anden rolle. Et eksempel på dette er tidspunktet for ordreafgivelse: I de deterministiske modeller kunne det præcist forudsiges, hvor meget lager der bruges



Figur 1

fra ordreafgivelse og til lageropfyldning; det kan vi ikke længere i det stokastiske tilfælde. Det er derfor generelt svært helt at undgå negativt lager. Vi skal antage, at der er en omkostning ved restordre, men at denne omkostning ikke afhænger af, hvor lang tid der går inden de kunder, der er på venteliste, kan tilfredsstilles.

En mulig udvikling af lageret over tid er vist i figur 1. Vi starter med en ordreafgivelse; derefter bruges der af lageret i perioden τ, som er den tid, der går fra ordreafgivelse til levering. Denne tid antages her fast; det kunne godt være en stokastisk variabel, men det ville gøre vor model mere indviklet.

Efter levering til lageret af mængden Q trækkes der stokastisk på dette lager indtil man når et tidspunkt, hvor der afgives ny ordre; således gennemløber lageret en cyklus begyndende med ordreafgivelse, med eventuelt negativt lager, efterfulgt af opfyldning, hvorefter lageret igen nedbringes. Vi benytter notationen T_i for varigheden af en bestemt (her den i'te) lagercyklus; T_i kan skrives som $T_i = T_i' + T_i''$, hvor T_i' og T_i'' er tiden med henholdsvis positivt og negativt lager.

Hvad er en lagerpolitik i denne situation? Faktisk afhænger det af omstændighederne. Vi skal antage, at det er muligt *løbende* at kontrollere lagerets størrelse, f.eks. fordi ethvert salg automatisk også registreres som træk på lageret (alternativet, periodisk inspektion af lageret, fører til en noget mere kompliceret model). Der bliver derfor ligesom i de deterministiske modeller to væsentlige handlingsparametre, nemlig et nedre niveau for lageret og en indkøbsordre.

Det er væsentligt at notere sig, at vi i den nuværende situation *ikke* uden videre kan bruge en faktiske lagerstørrelse som udslaggivende for ordreafgivelse: Hvis vi f.eks. afgiver ordre ved lagerstørrelse s på indkøb af Q enheder, kan det ske, at der efterfølgende er så stort et træk på lageret, at der er mere end Q i restordre, når de bestilte enheder Q ankommer efter ventetiden τ. I så fald er den oprindelige lagerpolitik gået i vasken, for lageret vil da altid være mindre end s.

Det er naturligvis ikke noget rigtig alvorligt problem, men snarere et krav om

at formulere tingene rigtigt: Vi definerer aktuel *lagerposition* som

$$\text{lager} + \text{enheder i ordre} - \text{restordre},$$

og det er dermed en minimal lagerposition r, som er vor handlingsparameter (sammen med Q).

Da lagerpolitikken skal fastlægges udfra et ønske om minimale gennemsnitlige omkostninger, har vi brug for nogle middelværdier knyttet til lagerpolitikken.

Betragt et forløb af lagerpolitikken over en (lang) periode t. Lad n_t være antal gennemførte lagercyklus'er i denne periode; vi har da, at forholdet

$$\frac{n}{t} = \frac{1}{Q}\,\frac{nQ}{t}$$

er en stokastisk variabel, som for store t på grund af de store tals lov konvergerer (i sandsynlighed) til λ/Q, idet den anden brøk netop er efterspørgsel pr. tidsenhed. Mere præcist betyder det, at

$$P(\{|\frac{n}{t} - \frac{\lambda}{Q}| > \varepsilon\}) \to 0$$

for alle $\varepsilon > 0$, og vi skriver det som

$$\operatorname{plim}_{t\to\infty} \frac{n}{t} = \frac{\lambda}{Q}.$$

Lad os dernæst indføre størrelsen P_{neg} som den del af den forløbne tid, hvor der har været negativt lager – eller, mere præcist, grænsen af forholdet mellem den tid ud af et samlet forløb på t, hvor lageret har været negativt, og t selv, dvs.

$$P_{neg} = \operatorname{plim}_{t\to\infty} \frac{\sum_{i=1}^{n} T_i''}{t}$$

Vi skal desuden bruge størrelserne

$$D = \operatorname{plim}_{t\to\infty} \frac{\sum_{i=1}^{n} \Omega_i}{t}$$

for gennemsnitligt antal enheder holdt på lager, idet Ω_i er antal stykår i den i'te lagercyklus (teknisk set er det integralet under kurven for lagerets samlede størrelse i det i'te gennemløb),

$$B = \operatorname{plim}_{t\to\infty} \frac{\sum_{i=1}^{n} \Delta_i}{t}$$

for gennemsnitlig restordre (her er Δ_i samlet antal stykår i restordre i i'te cyklus), og endelig

$$E = \operatorname{p\lim}_{t\to\infty} \frac{\sum_{i=1}^{n} \xi_i}{t},$$

der angiver det gennemsnitlige antal enheder solgt i en situation med restordre (ξ_i er kundeankomst i perioden med restordre i cyklus i).

Når disse størrelser spiller en væsentlig rolle, er det fordi det er rimeligt at sætte lagerpolitikkens omkostninger i relation til sådanne gennemsnitsværdier (det var forøvrigt også hvad vi gjorde i den deterministiske situation). Antager vi, at omkostningerne består af

- gennemsnitlige ordreomkostninger (b pr. ordre)
- gennemsnitlige lagerholdsomkostninger (a pr. stykår på lageret)
- gennemsnitlige omkostninger ved restordre (c pr. stykår i restordre)
- faste omkostninger ved hvert salg i restordre (d pr. ordre).

Med disse antagelser fås samlede omkostninger

$$\mathcal{K} = b\frac{\lambda}{Q} + aD(Q,r) + cB(Q,r) + dE(Q,r).$$

Her har vi fremhævet, at størrelserne B, D og E indført ovenfor afhænger af lagerpolitikken givet ved Q og r.

Næste skridt er at vælge Q og r således, at $\mathcal{K}$ minimeres. Men her kommer vi tydelig nok ingen vegne uden nærmere kendskab til størrelserne B, D og E og deres præcise afhængighed af lagerpolitikken. Dette er noget, der varierer med de antagelser, vi lægger på fordelingen af kundeankomster. Vi ser lidt på et specielt tilfælde i næste afsnit.

6. Poissonfordelt salg: Exakte udtryk for omkostning ved lagerpolitik

Vi vil i dette afsnit finde udtryk for størrelserne, som indgår i bestemmelsen af $\mathcal{K}$. Hertil vil vi (ligesom iøvrigt ved diskussionen af kømodeller i forrige kapitel), antage, at processen har forløbet længe, således at sandsynligheden for, at lageret har en bestemt størrelse, er uafhængig af det tidspunkt, hvor der observeres.

Som diskuteret ovenfor er det praktisk at bruge lagerposition som hjælpevariabel, og vi vil derfor bruge notationen $\pi(j)$ for sandsynligheden for lagerposition j. Når r bruges til at afgive ordre, følger det at lagerpositionen kan antage værdier fra $r+1$ til $r+Q$ (for hvis lagerpositionen var r, ville den straks være $r+Q$).

Betragt et lille interval Δt; i dette interval kan lagerpositionen bevæge sig fra $r+j$ til $r+j-1, j \geq 2$, hvilket vil ske hvis der sker en efterspørgsel. Dette sidste sker med sandsynligheden $\lambda\Delta t$ (da efterspørgslen følger en Poisson-proces). Hvis

$j = 1$ og der sker en efterspørgsel, rykker lagerpositionen som nævnt til $r+Q$, igen med sandsynligheden $\lambda\Delta t$. På grund af denne symmetri – enhver tilstand flytter sig til en bestemt næste med samme sandsynlighed – må de enkelte $\pi(r+j)$ være lige store, og da der er Q af dem, har vi

$$\pi(r+j) = \frac{1}{Q},$$

$j = 1, \ldots, Q$.

Nu kan vi desværre ikke nøjes med at betragte lagerpositionen, men må tilbage til det aktuelle lager, da det desværre er dette, der bestemmer omkostningerne. Vi indfører sandsynlighederne $\psi(x)$ for at der er netop $x \geq 0$ enheder på lager; igen antager vi, at disse sandsynligheder for store t er uafhængige af, på hvilket tidspunkt vi betragter lageret (noget som iøvrigt kan vises at være tilfældet, men det går ud over vores formåen).

For at finde $\psi(x)$ noterer vi os, at på et givet tidspunkt vil alt, som er i ordre i mere end τ tidsenheder også være leveret. Så hvis lagerpositionen var $r+j$ for τ tidsenheder siden, vil sandsynligheden for, at lageret har netop x enheder nu være lig sandsynligheden for, at der er efterspurgt $r+j-x$ enheder i løbet af denne periode (såfremt $r+j-x \geq 0$, 0 ellers). Vi har dermed

$$\psi(x) = \frac{1}{Q}\sum_{j=1}^{Q} p(r+j-x;\lambda\tau),$$

hvor $p(i,\nu)$ er punktsandsynligheden i Poisson-fordelingen,

$$p(i,\nu) = \frac{\nu^i}{i!}e^{-\nu}.$$

Lidt behændig omorganisering af leddene i summen ovenfor fører til det (noget) simplere udtryk

$$\psi(x) = \begin{cases} \frac{1}{Q}[P(r+1-x;\lambda\tau) - P(r+Q+1-x;\lambda\tau)], & 0 \leq x < r+1 \\ \frac{1}{Q}[1 - P(r+Q+1-x;\lambda\tau)] & r+1 \leq x \leq r+Q \end{cases}$$

hvor $P(i;\nu) = \sum_{j=1}^{i} p(i;\nu)$ er den kumulerede Poisson fordeling.

På tilsvarende måde kan vi indføre sandsynligheder $\eta(y)$ for, at der er y enheder i restordre (igen antaget tidsuafhængig for store t), og som ovenfor kan $\eta(y)$ udtrykkes relativt simpelt ved

$$\begin{aligned}\eta(y) &= \frac{1}{Q}\sum_{j=1}^{Q} p(y+r+j;\lambda\tau) \\ &= \frac{1}{Q}[P(y+r+1;\lambda\tau) - P((y+r+Q+1;\lambda\tau)].\end{aligned}$$

Vi kan nu nærme os de udtryk, der indgår i omkostningsfunktionen: Først noterer vi os, at P_{neg}, sandsynligheden for, at der ikke er positivt lager, kan skrives som

$$P_{neg} = \sum_{y=0}^{\infty} \eta(y) = \frac{1}{Q} \left[\sum_{u=r+1}^{\infty} P(u; \lambda\tau) - \sum_{u=r+Q+1}^{\infty} P(u; \lambda\tau) \right].$$

Hvis vi indfører udtrykket

$$\alpha(v) = \sum_{u=v+1}^{\infty} P(u, \lambda\tau),$$

kan vi kort skrive

$$P_{neg} = \frac{1}{Q}[\alpha(r) - \alpha(r+Q)].$$

Dermed har vi straks $E(Q, r)$, det gennemsnitlige antal restordrer, idet der gælder

$$E(Q,r) = \operatorname{plim}_{t\to\infty} \frac{\sum_{i=1}^{n} \xi_i}{t} = \operatorname{plim}_{t\to\infty} \left[\frac{\sum_{i=1}^{n} T_i''}{t} \frac{\sum_{i=1}^{n} \xi_i}{\sum_{i=1}^{n} T_i''} \right] = \lambda P_{neg}.$$

Vi får således

$$E(Q,r) = \frac{\lambda}{Q}[\alpha(r) - \alpha(r+Q)].$$

Det gennemsnitlige restordrebeholdning $B(Q, r)$ kan findes af

$$B(Q,r) = \sum_{y=0}^{\infty} y\eta(y)$$

$$= = \frac{1}{Q} \left[\sum_{u=r+1}^{\infty} (u-r-1)P(u;\lambda\tau) - \sum_{u=r+Q+1}^{\infty} (u-r-Q-1)P(u;\lambda\tau) \right].$$

Vi gør som ovenfor og indfører en hjælpestørrelse

$$\beta(v) = \sum_{u=v+1}^{\infty} (u-v-1)P(u;\lambda\tau),$$

og vi kan da skrive $B(Q, r)$ noget pænere som

$$B(Q,r) = \frac{1}{Q}[\beta(r) - \beta(r+Q)].$$

Vi mangler nu blot $D(Q,r)$, det gennemsnitlige lager. Her kan vi skyde en genvej, idet vi vender tilbage til udgangspunktet, nemlig lagerpositionen: Den gennemsnitlige lagerposition er kan skrives som en sum af

– gennemsnitlig lager

– gennemsnitlig ordre til leverance

minus

– gennemsnitlig restordre.

Vi har, at gennemsnitlig lagerposition er

$$\sum_{j=1}^{Q}(r+j)\pi(r+j) = \frac{1}{Q}\sum_{j=1}^{r}(r+j) = \frac{Q+1}{2} + r,$$

så vi får

$$D(Q,r) = \frac{Q+1}{2} + r - \lambda\tau + B(Q,r),$$

idet den gennemsnitlige mængde enheder i ordre svarer til gennemsnitlig efterspørgsel i ventetiden, altså $\lambda\tau$.

Vi har nu samtlige størrelser, som indgår i omkostningsudtrykket, og de kan nu sættes ind, hvorefter der minimeres. Det lader sig dog ikke gøre helt umiddelbart, idet udtrykkene er ganske komplicerede. Man kan da vælge enten at bruge en numerisk procedure, eller at simplificere udtrykkene en smule.

Et specialtilfælde. Som man kan se af det forrige, er det især de komplicerede udtryk for $E(Q,r)$ og $B(Q,r)$, som giver besvær. Her vil det hjælpe at se bort fra leddene $\alpha(r+Q)$ og $\beta(r+Q)$, noget som man kan gøre med sindsro når tiden τ er lille, således at det er meget usandsynligt, at efterspørgslen i denne periode vil overstige $r+Q$. Vi får da

$$\mathcal{K} = b\frac{\lambda}{Q} + a\left[\frac{Q+1}{2} + r - \lambda\tau\right] + d\frac{\lambda}{Q}\alpha(r) + \frac{1}{Q}(a+c)\beta(r)$$

Hvis enhederne i lageret er så små, at vi kan tillade os at tilnærme kontinuert, kan vi finde det optimale lager ved at differentiere og sætte lig nul. Denne procedure virker dog kun umiddelbart ved bestemmelsen af Q, som kan findes til

$$Q = \sqrt{\frac{2\lambda}{a}(b + d\alpha(r) + \lambda^{-1}(a+c)\beta(r)}.$$

For at finde den optimale værdi af r skal man benytte udtrykkene for $\alpha(r)$ og $\beta(r)$. Det skal vi ikke gå nærmere ind på.

7. Opgaver

1. En pizzabager skal beslutte sig for dagens produktion af pizzaer, der sælges i slices. Ingredienserne i en pizzaslice kan opgøres til kr.8,95, den vejer 100 g og sælges for 16 kr.

På grund af kommunens miljøregler må eventuelle usolgte slices transporteres til genbrugscentret, der ligger 12 km udenfor byen, og der skal her betales en komposteringsafgift på 38 kr.pr.kg. affald.

Køberne kommer erfaringsmæssigt jævnt over åbningstiden 8 – 16, og det gennemsnitlige antal købere er 26 i timen. Der er dog en del udsving: I knap en trediedel af den tid, bageriet har været åbent, har salget været over 40.

Hvor mange pizza-slices skal der bages?

8. Litteratur

Den deterministiske lagerteori er af ældre dato, idet der blev arbejdet med lagerstyringsmodeller allerede i 1920erne. Stokastiske lagermodeller, hvor man søger at fastlægge en optimal lagerpolitik under stokastisk efterspørgsel og/eller produktion, blev udviklet fra 1950erne og fremefter; et væsentligt bidrag blev givet i Arrow, Karlin og Scarf (1958). En gennemgang af denne teori (hvorfra også fremstillingen i afsnittene 5 og 6 er hentet) kan findes i Hadley og Whitin (1963).

KAPITEL 17

Køteori

1. Den generelle problemstilling; klassifikation af kømodeller

En almindelig forekommende situation, såvel fra gennemsnitsborgerens hverdag som i virksomheden, er den hvor en eller anden ydelse skal gives til kunder, forbrugere, afdelinger eller hvad det nu kan være, og hvor behovet for ydelsen er sådan, at der kan være flere, der samtidig ønsker betjening, således at der dannes en *kø*. Forekomsten af en sådan anses normalt for et negativt fænomen, men i en kvantitativ analyse, må dette negative aspekt naturligvis dels gives et konkret indhold, dels må det afvejes overfor omkostningerne ved at have kapacitet nok til altid at kunne betjene samtlige kunder.

Hvorledes denne afvejning rent faktisk skal finde sted, vil bero på, hvilke faktorer knyttet til det konkrete køfænomen, der er af størst betydning for den berørte virksomhed eller organisation, og det kan vi ikke sige noget generelt om. Hvad vi imidlertid kan gøre, er at rendyrke køproblematikken lidt mere for at finde nogle *centrale faktorer* ved en kø, noget som dels vil have betydning i en optimeringssituation, og som man dels vil kunne sige noget om udfra kendskab til problemets data. Herom handler *køteorien*. Vi skal i det følgende give en kort introduktion til denne teori.

Udgangspunktet er den ovenfor beskrevne situation: *Kunder* ankommer til betjening ved et eller flere *betjeningssteder; betjeningen* af den enkelte kunde *tager en vis tid*, og derfor vil der kunne opstå kødannelser: Der er ankommet flere kunder end svarende til de ledige betjeningssteder. Vi ønsker at beskrive den resulterende situation – hvor længe vil kunderne typisk skulle vente før de bliver betjent, hvor mange kunder vil på et givet tidspunkt stå i kø osv.

For at finde ud af sådanne ting må vi naturligvis vide lidt mere om det konkrete køsystem, og det er ret oplagt, at følgende forhold er centrale:

1. Kundernes ankomst
2. Betjeningstiden for kunder
3. Antallet af betjeningssteder.

Det vil derfor være oplagt at karakterisere en given kø ved de specielle omstændigheder under 1, 2 og 3, og det gør man faktisk også.

Ved næsten alle praktiske anvendelser vil det være således, at kunderne ikke kommer til bestemte, forud fastsatte tidspunkter, men at de kommer dryssende forholdsvis usystematisk: I relation til vor model betyder det, at vi modellerer *kundeankomsten* som stokastisk. Med den erfaring, vi har fra de foregående kapitler, vil det være naturligt, at kundeankomsten antages at følge en eller anden punktproces, der for hvert angivet tidsinterval beskriver sandsynligheden for ankomst af kunder i intervallet. Processen kan variere med den konkrete anvendelse, men det vil selvfølgelig være en fordel, hvis den er forholdsvis simpel, idet det ellers kan være svært at få præcise resultater ud. Vi vil nøjes med at betragte sådanne simple tilfælde.

På tilsvarende måde vil det være rimeligt at inddrage i vore overvejelser, at kundebetjeningen også vil være underkastet stokastik. Den enkelte kundes ekspedition afhænger af særlige omstændigheder ved såvel kunde som ekspedient, omstændigheder af en art, der dårligt lader sig inddrage på anden måde end ved beskrivelse som en stokastisk proces.

Til vort formål vil en kø – eller, for at være helt præcis, et køsystem, for der kan af og til være grund til at sondre mellem systemet med kunder, betjening osv. og selve køen, der kun består af de kunder, som *venter,* mens de betjente kunder jo ikke står i kø – være karakteriseret ved

(1) fordelingen for kundeankomst,
(2) fordelingen for ekspeditionstid, samt
(3) antallet af ekspeditionssteder.

Under visse omstændigheder kan der være behov for at tilføje detaljer, således f.eks. om øvre grænse for køens størrelse, men de tre nævnte forhold giver den grundlæggende beskrivelse af det system, man har med at gøre.

Denne måde at beskrive forskellige køtyper på blev indført af statistikeren D.G. Kendall (1953), der også lancerede en standardnotation, som er i almindelig brug. Først noterer vi os, at der er visse typer af fordelinger, som optræder tiere end andre i kømodeller (blandt andet fordi de har en naturlig baggrund i fænomener omkring stokastiske ankomster). Det drejer sig om

Eksponentialfordelingen M med tæthedsfunktion

$$f(x) = \frac{1}{A} e^{-x/A}, x \geq O$$

(bogstavet M som forkortet betegnelse skyldes at fordelingen især har vundet indpas efter en række arbejder af A.Markov),

Erlang fordelingen med parameter $k(E_k)$ har tæthedsfunktion

$$f(x) = \frac{b^{k+1} x^k e^{-bx}}{k!}, x \geq O$$

har to parametre (hvoraf kun k fremhæves explicit) med eksponentialfordelingens ene; fordelingen blev introduceret af den danske statistiker A.K.Erlang (ham vender vi tilbage til senere i kapitlet).

Deterministisk fordeling – betegnet D – har ikke nogen (sædvanlig) tæthedsfunktion, men fordelingsfunktionen er

$$F(x) = \begin{cases} 0 & x < a \\ 1 & x \geq a. \end{cases}$$

Generel (dvs. uspecificeret) *fordeling* betegnes med G.

Vi kan nu kort skrive en kø op som $\cdot/\cdot/\cdot$, hvor der på første plads står symbol for kundeankomstfordeling, på anden symbolet for betjeningsfordeling, og på sidste plads antal skranker. For eksempel er $M/E_k/7$ det køsystem, hvor kundernes ankomst er eksponentialfordelt, deres betjening Erlang-k-fordelt, og der er 7 luger til betjening.

2. Little's sætning

Der er ikke mange generelle resultater om køsystemer; så meget mere pusler man om det, der findes. Blandt de mest prominente er den såkaldte Little's sætning, der har været folklore blandt køteoretikere meget længe, men som første gang blev bevist af Little i 1951.

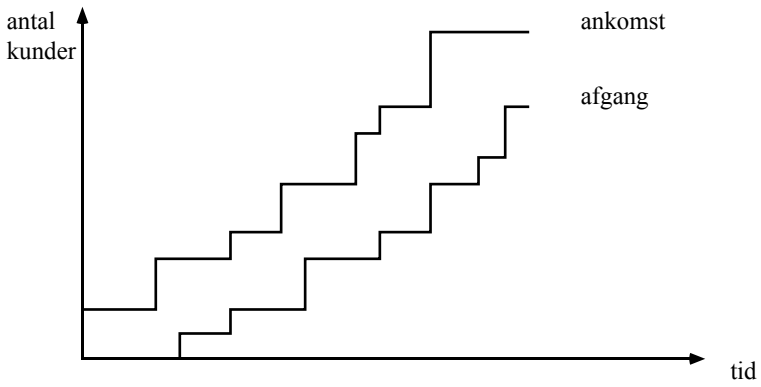
Little's sætning giver en sammenhæng mellem nogle basale variable, der beskriver forholdene i et køsystem. Hvis vi lader L være det gennemsnitlige antal kunder i systemet (der altså enten står i kø eller betjenes), λ den gennemsnitlige ankomstrate til systemet (nyankomne kunder pr. tidsenhed), og bruger W for den gennemsnitlige ventetid i systemet, siger sætningen, at

$$L = \lambda W.$$

Et intuitivt argument for Little's sætning er som følger: Vælg et tilpas stort T og lad $A(T)$ være antal ankomster indtil tidspunkt T, $B(T)$ den samlede ventetid i systemet præsteret af disse ankomne kunder. Man kan få en intuitiv fornemmelse for $A(T)$ og $B(T)$ ved at se på figur 1, hvor den øverste kurve angiver samlet tilgang af kunder over tid, altså $A(T)$, mens den nederste angiver samlet afgang af kunder. $B(T)$ er arealet mellem kurverne op til T; iøvrigt kan figuren også bruges til at aflæse antal kunder i systemet på et givet tidspunkt t (den lodrette afstand mellem kurverne) og ventetiden for hver given kunde (vandret afstand).

Vi definerer nu

$$\lambda(T) = \frac{A(T)}{T},\ W(T) = \frac{B(T)}{A(T)},\ L(T) = \frac{B(T)}{T},$$



Figur 1

hvor disse størrelser har den oplagte fortolkning som gennemsnit op til tidspunkt T. Der gælder

$$L(T) = \frac{B(T)}{T} = \frac{B(T)}{A(T)} \frac{A(T)}{T} = W(T)\lambda(T),$$

og hvis vi antager at størrelserne $\lambda(T)$, $W(T)$ og $L(T)$ går mod λ, W og L for $T \to \infty$, har vi det ønskede resultat.

3. Analyse af en kø; køens tilstand

Vi kan nu gå i gang med den detaljerede analyse af en kø; foreløbig starter vi generelt ud, hvilket med notationen fra forrige afsnit vil sige, at vi beskæftiger os med køen $G/G/k$; vi vil dog ikke her i starten gøre noget specielt ud af, at der er mange betjeningssteder, så vi antager $k = 1$.

De fænomener i forbindelse med køen, som har vor specielle interesse, er – som ovenfor anført – hvor mange kunder der gennemsnitligt befinder sig i køen, og hvor længe den enkelte kunde typisk skal vente (hvad der jo er forskellige aspekter af den samme sag). Vi har nu taget højde for den stokastiske karakter af vor kø ved at tilføje udtryk som "gennemsnitlig" og "typisk". Sådanne udtryk kan naturligvis gøres præcise: Hvad vi leder efter, er *middelværdien* af antal kunder og af ventetid.

Disse ting kan vi imidlertid ikke finde uden videre; ihvertfald skal vi først have passende fordelinger at tage middelværdier af. Her er det praktisk at indføre en slags hjælpevariabel, som vi vil kalde systemets *tilstand*. Hvad der nærmere skal forstås herved, gemmer vi et øjeblik, men formålet med tilstandsbeskrivelsen af systemet er, at vi med kendskab til systemets tilstand på tidspunkt t skal kunne forudsige hele den fremtidige udvikling lige så godt som hvis vi havde kendskab til hele forhistorien op til tidspunkt t. Tilstanden er altså en opsummering af køens forhistorie, detaljeret nok til, at ingen væsentlig information er gået tabt.

For at finde ud af, hvad der ihvertfald må indgå i beskrivelsen af systemtes tilstand, vil vi starte med at antage, at vi allerede *har* en sådan beskrivelse. Vi indfører nu notationen

$P_t(s) =$ sandsynligheden for at køen er i tilstand s på tidspunkt t,

$T_t(s, s', a) =$ sandsynligheden for at flytte sig fra tilstand s til tilstand s' i tidsintervallet fra t til $t + a$,

(hvor disse sandsynligheder er tætheder medmindre tilstandsrummet er diskret). Sammenhængen mellem disse er den oplagte, nemlig

$$P_{t+a}(s') = \sum_{s \in S} P_t(s) T_t(s, s', a)$$

(hvor vi igen skal erstatte $\sum$ med $\int$ hvis s varierer kontinuert). Ved hjælp af dette udtryk kan vi relatere sandsynlighedsfordelingen over tilstande på et givet tidspunkt til fordelingen på et andet tidspunkt. Det afgørende led her er naturligvis *overgangssandsynlighederne* $T_t(s, s', a)$. Disse må nødvendigvis have noget at gøre med ankomster og afgang fra køen i intervallet fra t til $t + a$.

Hvad angår *ankomster*, så skal vi her se nærmere på vor første sandsynlighedsfordeling (den som angiver sandsynligheden (eller rettere: tætheden)) $f(t')$ for at der går netop tiden t' inden der kommer en kunde. For små intervaller (altså lille a) skrevet som $\triangle t$ har vi, at sandsynligheden for en ankomst i intervallet $t + \triangle t$, *givet at der ikke er kommet nogen før* t, er

$$p(t, \triangle t) = \frac{f(t) \triangle t}{1 - F(t)},$$

hvor $F(t) = \int_o^t f(u) du$ er den kumulerede sandsynlighedsfordeling. Vi ser, at $p(t, \triangle t)$ *afhænger af* t, så for at beskrive køens tilstand udtømmende må vi kende dette tidspunkt.

I relation til det konkrete problem – kundeankomster – betyder det, at vi må vide, hvor lang tid der er gået, siden sidste kunde ankom (og vi dermed nulstillede vor registrering af næste kundeankomst). Et helt tilsvarende problem opstår, når vi går over til at se på *afgang fra køen*. I vor diskussion her vil vi antage, at kunderne kun går fordi de er blevet betjent (altså ikke fordi de er blevet pas på at vente). Når vi ser på køen på tidspunkt t, er en bestemt kunde under betjening. Der er givet en sandsynlighedsfordeling (med tæthed $g(t')$) for at betjeningen tager t'; der er imidlertid allerede gået en vis periode t, og med samme argumentation som tidligere ser vi, at sandsynligheden for, at betjeningen afsluttes i løbet af intervallet fra t til $t + a$ afhænger af, hvor længe kunden *allerede var betjent i forvejen*.

Ud af disse overvejelser har vi da, at systembeskrivelsen (i vor kø med kun en luge) må indeholde

1. Køens længde,

2. Den tid, som er gået siden sidste kundes ankomst.

3. Varigheden af de betjening, der er igang på det pågældende tidspunkt.

I et køsystem med flere luger skal pkt. 3 modificeres, idet vi nu må holde kontrol med varigheden af samtlige igangværende ekspeditioner. Det skal vi ikke gå nærmere ind på.

Med vor tilstandsbeskrivelse afklaret har vi nu alle ingredienser til at behandle køsystemer i detaljer. Der er blot den hage ved sagen, at de fleste systemer – naturligvis afhængig af specifikationen – er så komplicerede, at det ikke er muligt at finde eksplicite udtryk for de ønskede størrelser. Vi skal derfor nøjes med et særligt simpelt specialtilfælde, der ikke udmærker sig ved at være specielt udbredt i praksis, men hvor det dog er muligt at gennemføre den regnetekniske side af sagen. Den videre analyse – med mere interessante fordelinger – må man så finde i speciallitteraturen.

4. Køen $M/M/1$

I det specielle tilfælde, hvor *såvel* kundeankomst som kundebetjening er eksponentialfordelt (ikke nødvendigvis med samme parameter), bliver den generelle tilstandsbeskrivelse en del simplere. Det hænger sammen med, at den betingede fordeling for næste hændelse (ankomst af kunde, færdiggørelse af betjening) givet at der er gået tiden t siden sidste begivenhed, *ikke afhænger af* t. Med notationen fra sidste afsnit har vi nemlig, at

$$p(t, \triangle t) = \frac{f(t)\triangle t}{1 - F(t)} = \frac{1}{A}\,\frac{e^{-x/A}\triangle t}{e^{-t/A}}$$

idet

$$F(t) = \int_0^t \frac{1}{A} e^{-\frac{x}{A}}\, dt = 1 - e^{-\frac{t}{A}}.$$

For $\triangle t \to 0$ får vi, at $p(t, \triangle t)/\triangle t$ går mod

$$\frac{1}{A} e^{-\frac{1}{A}(x-t)},$$

hvilket netop er tætheden for eksponentialfordelingen for første kundeankomst efter tidspunkt t (ovenikøbet med samme parameter A.) Konsekvensen er, at det ikke længere er nødvendigt at holde styr på sidste kundeankomst og start af sidste betjening. Da tilstandsbeskrivelsen hermed bliver langt simplere, må det forventes at være tilsvarende nemmere at nå konkrete resultater, og det viser sig da også at holde stik.

Lad os først indføre *parameteren* $\lambda = 1/A$, som er kundernes ankomsthastighed (det er den inverse til middelværdien af den stokastiske variable "næste ankomst af en kunde", som er A, og tilsvarende for betjeningsfordelingen $g(x) = s^{-1}e^{-x/s}$, hvor vi indfører *parameteren* $\mu = 1/s$, hvor s er middelværdien i fordelingen. Vi kalder μ for betjeningshastigheden.

Da tilstanden er fuldt beskrevet ved antal ventende kunder, kan vi nøjes med at beskrive systemet ved $P_t(i)$, sandsynligheden for, at der er netop i kunder i systemet på tidspunkt t. Vi ser som tidligere på et lille interval $\triangle t$ fra t og fremefter, og vi har da

$$\begin{aligned} P_{t+\triangle t}(0) = {} & P_t(0) \times (\text{sandsynligheden for ingen hændelse i intervallet } \triangle t) \\ & + P_t(1) \times (\text{sandsynligheden for at en betjening fuldføres i } \triangle t) \\ & + P_t(2) \times (\text{sandsynligheden for at to betjeninger fuldføres i } \triangle t) \\ & + \ldots \end{aligned}$$

Vi kan her tillade os at se bort fra højere led; når $\triangle t$ går mod 0, vil leddet i tredie linie gå mod 0 lige så hurtigt som $(\triangle t)^2$, og generelt kan vi derfor skrive

$$P_{t+\triangle t}(0) = P_t(0)(1 - \lambda\triangle t) + P_t(1)\mu\triangle t + 0(\triangle t)^2,$$

idet sandsynligheden for en eksponentialfordelt hændelse i det lille interval $\triangle t$ netop er $\lambda\triangle t$, resp. $\mu\triangle t$. På helt tilsvarende måde kan vi skrive

$$\begin{aligned} P_{t+\triangle t}(j) = {} & P_t(j) \times \text{ (sandsynligheden for ingen hændelse)} \\ & + P_t(j-1) \times \text{ (sandsynligheden for en ankomst i } \triangle t) \\ & + P_t(j+1) \times \text{ (sandsynligheden for en fuldført betjening i } \triangle_t) \\ & + \text{ led som indeholder flere begivenheder i intervallet,} \end{aligned}$$

og dette kan tilsvarende omformes til

$$\begin{aligned} P_{t+\triangle t}(j) = {} & P_t(j)(1 - \lambda\triangle t)(1 - \mu\triangle t) \\ & + P_t(j-1)\lambda\triangle t + P_t(j+1)\mu\triangle t + O(\triangle t)^2, \end{aligned}$$

for $j = 1, 2, 3, \ldots$ Ved at flytte rundt får vi ligningssystemet

$$(P_{t+\triangle t}(0) - P_t(0))/\triangle t = \mu P_t(1) - \lambda P_t(0) + O(\triangle t),$$

$$\begin{aligned} (P_{t+\triangle t}(j) - P_t(j))/\triangle t = {} & \lambda P_t(j-1) + \mu P_t(j+1) \\ & -(\lambda + \mu)P_t(j) + O(dt), j = 1, 2, 3, \ldots, \end{aligned}$$

og for $\triangle t \to 0$ bliver det til systemet af differentialligninger

$$\frac{d}{dt}P_t(0) = \mu P_t(1) - \lambda P_t(0)$$

$$\frac{d}{dt}P_t(j) = \lambda P_t(j-1) + \mu P_t(j+1) - (\lambda+\mu)P_t(j), j = 1, 2, 3, \ldots.$$

Her er der uheldigvis mere end endelig mange ligninger, med det udelukker dog ikke, at man kan finde en løsning, dvs. en familie $\{P_t(j)|j = 1, 2, 3, \ldots\}$ af funktioner af t, som opfylder ligningerne. Det kan vises at løsningen har formen

$$P_t(j) = P_j + \sum_{k=1}^{\infty} A_{j,k}e^{-c_k t}, j = 1, 2, 3, \ldots,$$

hvor P_j, $A_{j,k}$, og c_k er konstanter. Det har vi nu ikke så meget brug for at vide, for vi skal interessere os mere for den langsigtede udvikling af køsystemet.

Ideen her er, at sandsynlighederne for at systemet på tidspunkt t befinder sig i den ene eller den anden tilstand (stadigvæk fuldt ud karakteriseret ved antallet af kunder i systemet, dvs. tallet j), for $t \to \infty$ vil gå mod en grænse. Man siger, at systemet vil bevæge sig mod en *steady state*. Eksistensen af en sådan kan man selvfølgelig ikke være sikker på fra starten, dvs. uden at kende systemets sandsynlighedsfordelinger. I vort tilfælde kan vi se af de generelle løsninger ovenfor, at den tidsafhængige del bliver mindre og mindre for voksende t, således at $P_t(j)$ for store t er bestemt alene ved det første konstantled P_j. Man kunne også argumentere sig frem til eksistensen af en grænse på anden måde.

Hvorom alting er, så snart vi ved, at der findes en grænse $P(j), j = 1, 2, 3, \ldots$, kan vi på ret simpel måde finde et udtryk for den. Vi har nemlig fra differentialligningssystemet ovenfor (hvor vi lader t gå mod uendelig) at der må gælde

$$\mu P(1) - \lambda P(0) = 0$$

$$\lambda P(j-1) + \mu P(j+1) - (\lambda+\mu)P(j) = 0, j = 1, 2, 3, \ldots,$$

hvoraf vi får, at der gælder

$$\mu P(j) = \lambda P(j-1), j = 1, 2, 3, \ldots,$$

så at $P(j) = (\frac{\lambda}{\mu})P(j-1) = (\frac{\lambda}{\mu})^j P(0), j = 1, 2, 3, \ldots$. Vi kan finde $P(0)$ fra betingelsen

$$\sum_{j=0}^{\infty} P(j) = 1 = \sum_{j=0}^{\infty} (\frac{\lambda}{\mu})^j P(0) = \frac{P(0)}{1-(\lambda/\mu)} \text{ hvoraf } P(0) = 1 - \frac{\lambda}{\mu},$$

for $(\frac{\lambda}{\mu}) < 1$ (idet vi har brugt summeformlen for en uendelig kvotientrække). Bemærk, at hvis denne betingelse ikke er opfyldt, da giver tallene $P(j), j = 0, 1, 2, \ldots$, ikke mening som en sandsynlighedsfordeling. Det er heller ikke så mærkeligt, for hvis $\lambda > \mu$ er kundehastigheden større end betjeningshastigheden, og hele systemet vil blive overbelastet. At der heller ikke er nogen steady state for $\lambda = \mu$ er måske mindre oplagt, men det kan vi altså se af det foregående.

Lad os se lidt nærmere på resultaterne for vor steady state (altså i tilfældet $\lambda < \mu$). Vi kan for det første se, at *sandsynligheden for, at der ekspederes en kunde* – eller, om man vil, den del af den samlede tid, hvor betjeningen er aktiv – vil være

$$1 - P(0) = \frac{\lambda}{\mu}.$$

I denne del af tiden vil der være en eller flere kunder i systemet; vi har dermed også, at *sandsynligheden for, at en kunde kommer til at vente*, er λ/μ.

Hvor længe en kunde kommer til at vente i køen, afhænger naturligvis af, hvor mange der er i den i forvejen. Hvis der ikke er nogen, vil kundens ophold i systemet være det samme som betjeningstiden, der var eksponentialfordelt. Hvis der imidlertid allerede står j kunder (hvoraf den første betjenes og de andre venter på at komme til) kan vi skrive ventetiden som

$$W_j = r_1 + s_2 + s_3 + \ldots + s_j$$

hvor r_j er den resterende betjeningstid for kunde 1 og s_i er betjeningstiden for kunde $i, i = 2, \ldots, j$. Nu så vi imidlertid ovenfor, at resterende betjeningstid var fordelt på nøjagtig samme måde som fuld betjeningstid (dette står og falder med, at vi har eksponentialfordelt betjeningstid), så W_j er summen af j uafhængige eksponentialfordelte variable; det kan vises, at W_j har tætheden

$$h(t) = \frac{\mu^j t^{j-1} e^{-\mu t}}{(j-1)!},$$

og det kan bruges til at finde sandsynligheden for at vente ialt t i køen; lad tæthedsfunktionen for denne ventetid være $W(t)$; vi har da

$$\begin{aligned} W(t) &= \sum_{j=1}^{\infty} P(j) W_j(t) = (1-\rho) \sum_{j=1}^{\infty} \frac{\rho^j \mu^j t^{-j-1}}{(j-1)!} e^{-\mu t} \\ &= (1-\rho) e^{-\mu t} \mu\rho \sum_{j=1}^{\infty} \frac{(\rho\mu t)^{j-1}}{(j-1)!} \\ &= (1-\rho) e^{-\mu t} \mu\rho e^{\rho\mu t} = \mu\rho(1-\rho) e^{-\mu(1-\rho)t}, \end{aligned}$$

hvor vi har sat $\rho = \frac{\lambda}{\mu}$. Det har specielt interesse at se på den *gennemsnitlige ventetid*, der kan findes som

$$\int_0^{\infty} t W(t) dt = \frac{\rho}{(1-\rho)\mu} = \frac{\rho}{(1-\rho)} s.$$

Endelig kan vi notere os, at det *gennemsnitlige antal kunder i systemet* kan findes som

$$\sum_{i=0}^{\infty} i P(i) = \frac{\rho}{(1-\rho)},$$

og det gennemsnitlige antal kunder som står i kø (og venter på at blive betjent), er

$$\sum_{i=1}^{\infty}(i-1)P(i) = \frac{\rho^2}{(1-\rho)}.$$

Vi har dermed et relativt godt detailkendskab til vort køsystem, ihvertfald på længere sigt. Her skal det selvfølgelig tages i betragtning, at det system, som vi har studeret, også var særdeles simpelt – og nok heller ikke så realistisk; bag anvendelsen af eksponentialfordelingen ligger en antagelse om, at sandsynligheden for ankomst af kunde og afslutning af betjening i et lille interval slet ikke afhænger af forhistorien. Det passer givetvis dårligt på mange typer af køsystemer, specielt hvad angår betjeningsdelen. I sådanne tilfælde må der benyttes andre fordelinger, og teorien bliver da mere kompliceret.

Videre bør vi naturligvis udvide analysen til at dække køsystemer med flere betjeningsluger. Her kan der komme en overvejelse ind omkring *kødisciplinen*: Står kunderne i en lang række, hvorfra man altid kalder den første hen til en ledig luge (first-come-first-served), eller bruger man andre regler? Endelig burde man inddrage det forhold, at en kunde, der ankommer og ser den alenlange kø, beslutter sig for slet ikke at stille sig op i den ("balking"). Som man ser, er der emner nok tilbage, og det er ikke underligt, at litteraturen om køteori er meget omfattende. Vi har kun givet en lille smagsprøve.

5. Opgaver

1. For at mindske utilfredsheden hos kunder, der venter på betjening, har en pølsemand indført den regel, at hvis der er mere end tre kunder i kø (foruden den kunde, der aktuelt betjenes), får kunden gratis to ristede med brød (til kr. 22.50). Tilbudet gælder i frokostperioden fra 11-14, hvor der i gennemsnit kommer en kunde hvert andet minut, mens betjeningen tager 75 sekunder.

Hvor stor er sandsynligheden på et givet vilkårligt tidspunkt for at dette tilbud kommer i brug? Angiv forudsætningerne.

6. Litteratur

Køteori er et område af operationsanalysen – og mere generelt, af anvendt matematik – som vi ikke kan yde tilnærmelsesvis fuld retfærdighed; litteraturen på området er idag særdeles omfattende. Som selvstændig disciplin kan køteorien føres tilbage til arbejder af Erlang (1909) om køproblemer knyttet telefoncentraler (Erlang var ansat i KTAS). De matematiske aspekter af køteorien blev udforsket fra 1930 og fremefter med blandt andet papirer af Kolmogorov (1931) og Khintchine (1932).

Litteratur

Ackoff, R.L., and M.W.Sasieni (1968), Fundamentals of Operations Research, Wiley, New York.

Arrow, K.J., S.Karlin og H.Scarf (1958), Studies in the mathematical theory of inventory and production, Standford University Press, San Francisco.

Ashcroft, H. (1950), The productivity of several machines under the care of one operator, Journal of the Royal Statistical Society B 12, 145.

Ascher, H., and H.Feingold (1984), Repairable systems reliability: modeling, inference, misconceptions and their causes, Marcel Dekker, New York.

Bellman,R. & S.Dreyfus (1962), Applied Dynamic Programming, Princeton University Press, Princeton N.J.

Borovkov, A.A. (1986), Teorija Verojatnostej, Nauka, Moskva (russisk).

Charnes, A., W.Cooper, E.Rhodes (1978), Measuring the efficiency of decision making units, European Journal of Operational Research 2, 429 – 444.

Chartrand, G. (1985), Introductory Graph Theory, Dover Publications Inc., New York.

Churchman, C., R.L.Ackoff, and E.L.Arnoff (1957), Introduction to Operations Research, Wiley, New York.

Christofides, N., A.Mingozzi og P.Toth (1979), The Vehicle Routing Problem, Kap.11 i: Combinatorial Optimization, Wiley, New York.

Cohen, S.S. (1985), Operational Research, London.

Conway,R.W., W.L.Maxwell og L.W.Miller (1967), Theory of scheduling, Addison-Wesley, Reading, Massachusetts.

Erlang, A.K. (1909), Sandsynlighedsregning og telefonsamtaler, Nyt Tidsskrift for Matematik B 20, 33.

Feller, W. (1950), An Introduction to Probability Theory and Its Applications I, Wiley, New York.

Feller, W. (1966), An Introduction to Probability Theory and Its Applications II, Wiley, New York.

Fischer, M.L. og R.Jaikumar (1981), A generalized assignment heuristic for vehicle routing, Networks 11, 109 – 124.

Ford, L.R., and D.R.Fulkerson (1967), Flows in networks, Princeton University Press, Princeton, New Jersey.

Färe, R., S.Grosskopf og C.A.K.Lovell (1984), The measurement of efficiency of production, Kluwer-Nijhoff, Boston.

Gale,D. (1960), The theory of linear economic models, McGraw-Hill Publishing Company.

Garey.M.R. and D.S.Johnson (1979), Computers and intractability, a guide to the theory of NP-completeness, W.H.Freeman and Company, San Francisco.

Gillett, B. og L.Miller (1974), A heuristic algorithm for the vehicle dispatch problem, Oper.Res. 22, 340 – 349.

Goddard, L.S. (1963), Mathematical Techniques of Operational Research, Pergamon Press, Oxford.

Hadley, G., og T.M.Whitin (1963), Analysis of inventory systems, Prentice-Hall, Englewood Cliffs, New Jersey.

Harary, F. (1990), Graph Theory, Addison-Wesley, Reading, Massachusetts.

Hiller,F.S. and G.J.Lieberman (1995), Introduction to Operations Research, McGraw-Hill Publishing Company, 6.ed.

Johnson, L.A. (1974), Operations research in production planning, scheduling, and inventory control, Wiley, New York.

Karmarkar, N. (1984), A new polynomial-time algorithm for linear programming, Combinatorica 4, 373 – 395.

Khachian, L.G. (1979), A polynomial Algorithm for linear programming, Dokl. Akad. Nauk USSR 244, 5 (russisk), engelsk overs. i Soviet Math. Doklady, 20, 1991-94.

Khintchine, A.Ja. (1932),Ẅathematisches über die Erwartung vor einem öffentlichen Schalter, Mat.Sbornik 39, 151.

Kolmogorov,A. (1931), Sur le problème d'attente, Mat.Sbornik 38, 101.

Kuhn, H.W. (1955), The Hungarian method for the assignment problem, Nav.Res. Logistics Quarterly 2, 83 – 97.

König, D. (1950), Theorie der endlichen und unendlichen Graphan, Chelsea, New York.

Lagarias,J.C. and M.J.Todd (eds.) (1990), Mathematical developments arising from linear programming, American Mathematical Society.

Lawler, E.L., J.K.Lenstra, A.H.G.Rinnooy Kan og D.B.Shmoys (1985), The Traveling Salesman Problem, Wiley, New York.

McHugh, J.A. (1990), Algorithmic Graph Theory, Prentice-Hall International, Englewood Cliffs, New Jersey.

Nemhauser, G.L. and L.A.Wolsey (1988), Integer and Combinatorial Optimization, John Wiley and Sons, New York.

Papadimitriou, C.H. and K. Steiglitz (1982), Combinatorial Optimization: Algorithms and Complexity, Prentice-Hall, Englewood Cliffs, New Jersey.

Schrijver, A. [1986], Theory of Linear and Integer Programming, John Wiley and Sons, New York.

Shor, N.Z. (1970), Utilization of the operation of space dilation in the minimization of convex functions, Kibernetika 6, 6-12 (russisk), engelsk overs. i Cybernet-

ics, 6, 7-15.
Simmons, D.M. (1975), Nonlinear programming for operations research, Prentice-Hall, Englewood Cliffs, New Jersey.
Taha, H.A. (1997), Operations Reseach - An Introduction, Prentice-Hall.
White,D.J. (1969), Dynamic Programming, Holden-Day, San Francisco.
Wolfe, P. [1959] The simplex method for quadratic programming, Econometrica 27, 382 – 398.
Zangwill, W.I. [1967], The convex simplex method, Management Sci. 14, 231 – 238.

Stikordsregister

acyklisk graf 124
afrunding 84
afstandsklasser 126
assignment 183
aviskioskmodellen 263
baglæns markering 195
basis 20
beregningskompleksitet 214
betjeningsfaktor 246
betjeningssteder 274
Blackwell's sætning 239
branch-and-bound 93
busparadokset 242

CMT 2-fase algoritmen 110
cykel 123

Dantzig-Wolfe dekomponering 41
DEA 44
decisionsproblem 216
defekt 241
degenererede løsninger 178
Dinic's algoritme 150
dual simplex 35
dualitet (i lineær programmering) 19
dynamisk programmering 113

Edmonds-Karp algoritmen 142
eksponentialfordeling 275
ellipsoid-algoritmen 51
Erlang-fordeling 275
excess 241

Farrell-index 48
farvning af graf 128
flow conservation 136
flow-shop problemet 207
foldning 256
forlæns markering 194
fornyelsesprocesser 236
fractional cut 89

Gantt-kort 191
genanskaffelse 233
grad 121
graf, definition 118
gyldige uligheder 83

Hall's ægteskabssætning 155
Hamilton-cykel 124

ikke-deterministisk Turing maskine 222

Johnson's algoritme 209

kant (i graf) 118
kantsnit 130
kapacitet (af netværk) 135
kapacitet (af snit) 139
Karmarkar's algoritme 56
kilde 135
klargøring (af tableau) 27
klarperiode 243
knapsack problem 79, 116
konveks simplex 71
kostplan-problemet 15
kredsløb 122
kritisk vej 194
Kuhn's metode 185
kvadratisk programmering 62
kvadratrodsformlen 261
kø 274
König's sætning 154, 184

lager 259
Lagrange-dualitet 89, 169
lineær programmering 13
Little's sætning 276

markeringsproces 144
maskinproblemet 243
max-flow-min-cut 141

netværk 135
NP-komplethed 225
NP-problemer 223

operationsanalyse, definition 9
orienteret graf 120
out-of-kilter metoden 196
overdækning 152

parring 152
pivotstep 24
plan graf 131
Poisson-processen 232
polynomiale problemer 221
projekt 191
punktduelighed 254
punktprocesser 232
punktsnit 130
pålidelighedssystemer 250

regulær graf 121
relaxation 82
reparationsperiode 243
revideret simplex 37

sammenhængende graf 124
simplex-algoritmen 20
simplex-tableau 21
skæringsplanalgoritmer 89
slackvariable 17
snit (i netværk) 138
strukturfunktion 251
strukturrang (af matrix) 184
strøm (i netværk) 137
sweep algoritmen 110

terminal 135
todelt graf 152
transportproblem 174
træ 124
TSP (travelling salesman problem) 100
Turing maskine 218

ULP 166

vej 123
VRP (vehicle routing problem) 107

Wolfe's algoritme 62