Grafteori og optimering
en kort innføring

Geir Dahl*

24. oktober 2001

Innhold

Forord

Denne rapporten er skrevet med henblikk på et etter- og videreutdanningskurs for matematikklærere i den videregående skolen.

Man regner at grafteori oppsto i 1736 i byen Königsberg. Denne starten ble preget av den kjente matematikeren Leonard Euler. Det var ikke bare grafteori som oppsto der og da, men også fagfeltet kombinatorisk optimering (som, noe forenklet, er studiet av optimeringsproblemer i grafer).

Dette heftet gir en kort innføring i grafteori og optimering i grafer. Dette er relativt unge disipliner i matematikk som er i kraftig vekst. Dette skyldes ikke minst at grafbegrepet er av generell art og er egnet for modellering av mange praktiske situasjoner. I tillegg er optimering jo "matematikken for å finne optimale løsninger" og dette passer godt sammen med grafbegrepet. I denne rapporten er det derfor forsøkt å gi eksempler på hvordan en del praktiske problemer og anvendelser passer inn denne rammen.

Jeg har et håp om at kursdeltakerne kan se muligheter for å hente inspirasjon i denne teksten til f.eks. prosjektarbeid i matematikk i videregående skole. I samspillet mellom matematiske begreper og tenkning og anvendte problemer ligger et stort og flott potensiale, både for anvendt matematisk forskning og for å stimulere interessen for matematikk i sin alminnelighet.

Universitetet i Oslo Oslo 5. juni 2001

Geir Dahl

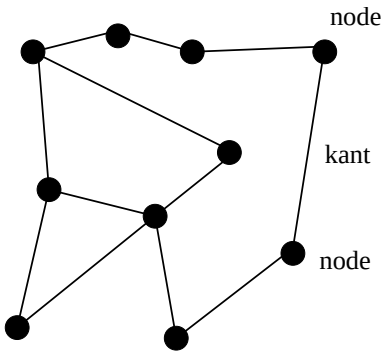
Kapittel 1

Introduksjon til grafteori

Dette kapitlet inneholder en kort introduksjon i de forste begrepene i grafteori. I tillegg nevnes noen eksempler på sammenhenger der grafer naturlig dukker opp.

1.1 Hva er en graf?

Vi er omgitt av grafer! Ta, for eksempel, et kart over gatene i en by, si Oslo. Her finner vi gater og gatekryss. Hver gate forbinder to gatekryss. Et slikt kart er et eksempel på en graf: vi har visse gatekryss, disse kalles *noder* i grafen og vi hare visse gater, disse kalles *kanter* i grafen. En kant er altså en forbindelse mellom to noder.



Figur 1.1: En graf som viser et veinettverk.

Et kart brukes jo til å finne fram i byen. Hvis vi befinner oss på Stortorget og vil finne veien til Nils Henrik Abels hus på Blindern, tar vi en kikk på kartet. Vi leter oss fram til en vei fra S til NHA. En slik vei består av påfølgende gater

som forbinder S og NHA. Slik er det i grafer også: vi har der begrepet vei som er en sekvens av påfølgende kanter som forbinder to noder. Mange viktige og interessante problemer i grafer har med veier å gjøre. La oss nevne et par slike:

1. Hvis en graf er gitt, hvordan kan vi finne en vei mellom to gitte noder, eller eventuelt vise at ingen slik vei finnes?
2. Hvordan kan vi finne en kortest mulig vei mellom to noder, si S og NHA? Her tenker vi oss at det er gitt en viss lengde knyttet til hver kant. Dette kalles *korteste vei problemet.*

Vi skal i dette kurset se nærmere på disse og andre beslektede spørsmål. Dette innebærer at vi skal:

- Definere en del grafbegreper og studere disse i noen grad.
- Se på ulike problemer det er naturlig å se på i grafer. Noen slike problemer dreier seg om *eksistens*, altså hvorvidt et objekt eller en viss egenskap er tilstede i grafen. Andre slike problemer involverer optimering.
- Mange problemer i grafteori er av en slik art at man ikke kan skrive ned en løsning eksplisitt eller på en enkel måte. Derimot kan man finne en prosedyre som skritt for skritt finner fram til den ønskede løsningen. En slik prosedyre kalles en *algoritme*. Vi skal se på noen algoritmer i dette kurset.
- Vi skal se på slike algoritmer fra en matematikers ståsted. Dette innebærer at vi er opptatt av matematisk innsikt i problemet og hvordan dette leder til en algoritme. Å forstå en algoritme og hvorfor den virker kan være både interessant og artig.

Vi gir et nok et eksempel til på en graf i en praktisk situasjon.

Eksempel: La oss se på alle innbyggerne i Fredrikstad, det er som kjent N innbyggere! Vi lager oss en graf G som har en node for hver innbygger. Grafen har også kanter (ellers ville den ikke være så interessant); det er en kant mellom to personer hvis de kjenner hverandre. (To gitte personer vil enten kjenne hverandre eller ikke, "nesten-kjennskap" ser vi bort fra her). Nå kan vi stille flere interessante spórsmål, f.eks.:

- Betrakt en gruppe mennesker der alle kjenner alle; la oss kalle en slik gruppe for en *klikk*. Hva er den største klikken i Fredrikstad, hvor stor er den? Og hvordan kan vi finne en slik klikk? Tja, ..
- Vi kan også se på "det motsatte", nemlig en gruppe mennesker der ingen kjenner hverandre. Hva er en størst mulig slik gruppe og hvordan kan vi finne den?
- Betrakt en gruppe S av personer som er slik at hver av de N innbyggerne kjenner minst én person i S. Kan vi finne en slik mengde S med f.eks. 20

personer. Eller, enda bedre, med færrest mulig personer (i S).

- Vi kan definere *avstanden* mellom to personer som det laveste antall kanter i en vei mellom de to personene (sekvens med suksessive bekjente). Hva er den største avstanden mellom to personer i Fredrikstad.

Vi skal komme tilbake til noen av disse spørsmålene senere. □

Optimering. Som nevnt skal vi også studere optimering i forbindelse med grafer. Matematisk optimering er den delen av matematikk der man studerer minimum og maksimum av funksjoner. En typisk situasjon er at man har gitt en viss mengde A og en reell-valuert funksjon f definert på A, så for hver $x \in A$ er $f(x)$ et reellt tall. Problemet er å finne en $x_0 \in A$ som har minst mulig funksjonsverdi, dvs. som oppfyller at

$$f(x_0) \leq f(x) \quad \text{for alle } x \in A.$$

Vi kaller da x_0 en *optimal løsning*. Hvor komplisert et slikt problem er henger naturligvis sammen med hva slags mengde A er og egenskaper til funksjonen f. Elementene i A kalles *tillatte løsninger*. Betegnelsen henger sammen med at oftest er A en viss delmengde av en større mengde og det er bare elementene i A som er "tillatte" i det aktuelle problemet.

Vi skal etterhvert se på to typer optimeringsproblemer:

- I den ene typen problem er A en endelig mengde og f er en enkel ("lineær") funksjon. Utfordringen ligger i at A har så mange elementer, altfor mange til at vi kan sammenlikne alle sammen og finne en optimal løsning på en slik direkte måte. Slike problemer studeres i området *diskret optimering* eller *kombinatorisk optimering*; dette er en egen retning i fagfeltet optimering.
- Den andre typen optimeringsproblemer er spesielle lineær programmeringsproblemer. Lineær programmering (LP) er å finne maksimum (eller minimum) av en lineær funksjon i n reelle variable under begrensninger gitt ved visse lineære likninger og/eller lineære ulikheter. LP er et svært viktig felt innen optimering: her finnes viktig teori, effektive algoritmer og betydningsfulle anvendelser på en lang rekke områder. Vi vil altså se nærmere på en spesielle type LP problemer som har tilknytning til grafer.

1.2 Noen grunnleggende begreper

Vi vil nå definere en graf mer presist, introdusere noen nyttige begreper og studere enkelte grunnleggede egenskaper.

En graf består altså av to typer objekter: noder og kanter. En graf er et (ordnet) par $G = (V, E)$ der V og E er endelige mengder (med en viss relasjon). V kalles *nodemengden* og elementene kalles *noder* (eller *hjørner* eller *punkter*). E kalles *kantmengden*, og her er hvert element faktisk en delmengde av V bestående av to noder. Hvert element i E kalles en *kant* (eller *linje*). Alstå: en graf består av en endelig nodemengde og visse par fra nodemengden. (Notasjonen V og E er vanlig og henspeiler på de engelske termene *vertex* og *edge*). En kant e bestående av nodene u og v kan betegnes $e = \{u, v\}$, alstå vanlig mengdenotasjon. Husk at $\{u, v\} = \{v, u\}$; rekkefølgen i mengdebeskrivelser spiller ingen rolle. Ofte brukes også notasjone $[u, v]$ eller bare uv, og vi vil stort sett benytte disse i denne rapporten.

Det er viktig å merke seg at begrepet graf er abstrakt, det involverer abstrakte, endelige mengder. Likevel kan en graf gjøres til noe svært konkret i form av en tegning i planet. Man tegner da et punkt eller en liten sirkel for hver node slik at disse sirklene ikke snitter. Deretter representeres hver kant ved en kurve mellom de to aktuelle sirklene, som oftest bruker man et linjesegment.

Eksempel: Betrakt en graf $G = (V, E)$ der $V = \{v_1, v_2, v_3, v_4\}$ og E består av kantene $e_1 = [v_1, v_2]$, $e_2 = [v_2, v_3]$, $e_3 = [v_1, v_3]$ og $e_4 = [v_1, v_4]$. Dette er en graf med 4 noder og 4 kanter. I figuren under vises to ulike representasjoner av denne grafen i planet. En slik representasjon kalles en *embedding* av grafen i planet eller, enkelt og greit, en tegning av grafen. □

Figur 1.2: Ulike embeddinger.

For å kunne snakke (eller skrive) rimelig enkelt om grafer er det en del betegnelser man velger å bruke. Betrakt en kant $e = [u, v]$. Vi sier at e er en kant *mellom* u og v, eller at e *forbinder* u og v. Videre kalles da u og v for *endenodene* til e. Enhver kant har altså nøyaktig to endenoder. Hvis det er en kant mellom to noder, så kalles disse nodene for *naboer*. I grafen i Figur 1.2 er f.eks. v_1 og v_2 naboer, kanten e_2 går mellom nodene v_2 og v_3, og noden v_4 har bare v_1 som nabo.

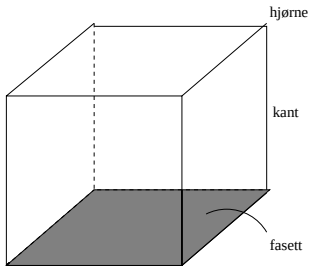
Grafer og polyedre. Vi har nevnt at man av og til bruker betegnelsen "hjørne"

istedet for "node". Hvorfor brukes betegnelsene "hjørne" og "kant" om de grunnleggende grafobjektene? Dette henger sammen med at grafer på midten av 1800-tallet ble studert i tilknytning til polyedre. Et polyeder er en viss type geometrisk objekt; de punktene i f.eks. rommet som oppfyller et system av lineære ulikheter. Noen eksempler er kuber og tetraedre. Når man studerer polyedre, er det visse deler av randen som har spesiell interesse, dette er *sider* i polyedret. For et polyeder i rommet (altså i $\Re^3$) har man tre typer (ikketrivielle) sider: *hjørner*, *kanter* og *fasetter*, se Figur 1.3. Her er hjørne, kant og fasett sider av dimensjon henholdsvis 0, 1 og 2.

Til ethvert polyeder kan vi knytte en graf. Betrakt f.eks. kuben P i Figur 1.3 og la G være grafen med 8 noder, en node for hvert hjørne, og med en kant mellom to noder hvis de tilhørende hjørnene er endepunktene til en kant i P. Denne grafen er tegnet i Figur 1.4. og den har 12 kanter. Legg merke til at hver node har grad 3, så dette er eksempel på en *regulær graf*; alle noder har samme grad. Vi ser også at antall fasetter er 6. Dermed er

$$\text{(antall noder) - (antall kanter) + (antall fasetter)} = 2.$$

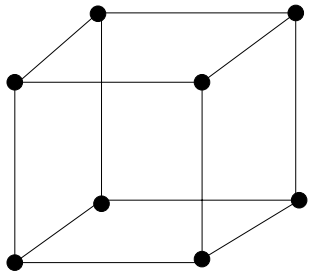
Dette er ingen tilfeldighet! Denne formelen kalles *Euler's polyederformel* og holder for alle polyedre (av dimensjon 3) i rommet. Faktisk kan denne formelen utledes ved å studere planare grafer (grafer som kan tegnes i planet uten kryssende kanter).



Figur 1.3: Kuben: ulike sider.

Om definisjonen av graf. Slik vi har definert en graf vil to distinkte (ulike) kanter ikke kunne ha de samme endenodene. Imidlertid har man av og til behov for å tillate slike *parallelle kanter*. En annen variasjon er å tillate at begge endenoder for en kant faller sammen (slik at kanten er en delmengde med bare ett element fra nodemengden). En slik kant kalles en *løkke*. Men, med den definisjonen vi her bruker, vil en graf ikke ha paralleller eller løkker. En graf med paralleller og løkker kalles gjerne en *multigraf*.

To parametre er spesielt mye brukt for grafer:



Figur 1.4: Grafen til kuben.

- $|V|$: antall noder, betegnes heretter med n og kalles *ordenen til grafen*
- $|E|$: antall kanter, betegnes heretter med m.

Eksempel: Betrakt en graf med 4 noder der hvert par av noder er naboer. Vi betegner denne grafen med K_4 og den kalles den *komplette* grafen med 4 noder. Den har 6 kanter. Mer generelt, i den komplette grafen med n noder er hvert par av noder naboer, og grafen betegnes med K_n. □

Likhet av grafer. K_n kalles den komplette grafen med n noder. "Den"? Kan det ikke være flere komplette grafer med n noder? Jo, men dette henger sammen med hva vi mener med at to grafer er like. Det finnes to slike begreper. Det strengeste begrepet er å si at to grafer er like dersom nodemengdene er like og kantmengdene er like, dvs. $G = (V, E)$ og $G' = (V', E')$ er like betyr at $V = V'$ og $E = E'$. F.eks., la G være gitt ved $V = \{u, v\}$ og $E = \{e\}$ der $e = [u, v]$. Da er det bare én graf som er lik G, nemlig grafen med nodemengde $\{u, v\}$ og kantmengde $\{e\}$ der stadig $e = [u, v]$. Men hva med grafen G' som har de to nodene p og q og har én eneste kant, nemlig $f = [p, q]$? G og G' har jo "samme struktur": to noder med en kant mellom disse. Vi sier at G og G' er *isomorfe*, som da er vårt andre likhetsbegrep. Løst sagt er to grafer isomorfe hvis det bare er navnene vi har valgt på noder og kanter som skiller dem. Alternativt: vi kan tegne de to grafene i planet på en slik måte at de faller sammen. Den presise definisjonenen er som følger: grafene $G = (V, E)$ og $G' = (V', E')$ er *isomorfe* dersom det fins en bijektiv funksjon $f : V \to V'$ slik at to noder u og v er naboer i G hvis og bare hvis $f(u)$ og $f(v)$ er naboer i G'. Husk her at en bijeksjon er en énentydig funksjon (dvs. distinkte elementer avbildes på distinkte elementer) som også er surjektiv (dvs. ethvert element i V' er bildet av et passende element i V).

Isomorfe grafer har nøyaktig de samme egenskapene, når vi da tenker på egenskaper som ikke er knyttet til selve navngivingen av graf objektene (noder, kanter). For små grafer kan vi for hånd raskt finne ut om to grafer er isomorfe. For større

grafer blir dette for mye arbeid, men det finnes algoritmer slik at en datamaskin kan gjøre jobben for oss. Imidlertid finnes det ingen kjente *raske* algoritmer for dette problemet: selv for grafer av moderat størrelse vil algoritmene ta for lang tid til at de har noen praktisk betydning. I en viss teoretisk forstand er det derfor et vanskelig problem å avgjøre om to gitte grafer er isomorfe. Dette er et såkalt *NP*-komplett problem. Vi skal vende noe tilbake til algoritmer og kompleksitet senere.

I en graf $G = (V, E)$ er det naturlig å se på hvor mange naboer hver node har. Vi definerer *graden* til en node v som antall naboer til v, dvs. antall kanter som er inntil v. Denne størrelsen betegnes med $d(v)$, eller $d_G(v)$ hvis det er behov for å angi hvilken graf det er snakk om.

Til hver graf $G = (V, E)$ med n noder har vi altså de n heltallene som er gradene til nodene. Har disse tallene bestemte egenskaper? Javisst, og vi kan etablere den mest grunnleggende egenskapen ved et lite telleargument. Vi vil bestemme summen av alle gradene til nodene, dvs.

$$\sum_{v \in V} d(v).$$

Når vi beregner denne summen vil vi for hver node v telle opp antall kanter inntil denne noden. Men dette må bety at i denne opptellingen vil hver kant bli talt opp nøyaktig to ganger, én gang for hver av de to endenodene. Derfor må summen av gradene bli lik to ganger antall kanter i grafen. Vi har altså følgende resultat, det kalles *grafteoriens første teorem*. Det er å finne i den første artikkelen i grafteori; den ble skrevet av Leonard Euler i 1736.

Teorem 1.1. *For enhver graf $G = (V, E)$ har vi at*

$$\sum_{v \in V} d(v) = 2m$$

Følgelig vil G ha et like (partall) antall noder av odde grad.

Gradsekvensen til en graf $G = (V, E)$ av orden n (dvs. n noder) er vektoren

$$d = (d_1, d_2, \ldots, d_n)$$

der $d_1, d_2, \ldots, d_n$ er gradene til nodene ordnet i ikkevoksende rekkefølge, dvs. $d_1 \geq d_2 \geq \cdots \geq d_n$. F.eks. er gradsekvensen til K_n lik $(n-1, n-1, \ldots, n-1)$.

Nå vil vi definere det viktige begrepet "vei" i en graf $G = (V, E)$. Ha innledningseksemplet med et veikart i tankene! En *vei* i G er en sekvens P av påfølgende kanter

$$P : [v_0, v_1], [v_1, v_2], \ldots, [v_{k-1}, v_k]$$

der nodene $v_0, v_1, \ldots, v_k$ er distinkte (og alle kantene er i G). Vi sier at veien P har *lengde* k (antall kanter) og at P er en vei mellom v_0 og v_k. Man sier også at v_0 og v_k er *endenoder* for P og at nodene $v_1, v_2, \ldots, v_{k-1}$ er *indre noder* i veien. En vei av lengde k har altså k kanter og den "inneholder" $k+1$ noder.

Hvor mange veier er det mellom to gitte noder i en graf? Det finnes ikke noe enkelt svar på dette, unntatt hvis grafen har "pen" struktur. Men det finnes algoritmer som kan suksessivt finner alle veiene og dermed får man også vite hvor mange slike veier grafen inneholder. Ofte vil antall veier mellom to noder være gigantisk. Et slikt eksempel ser vi nærmere på.

Eksempel: Betrakt den komplette grafen K_n, si med noder $v_1, v_2, \ldots, v_n$. Hvor mange veier er det mellom v_1 og v_n? (Det må være like mange veier mellom hvert par av noder). La N betegne dette antallet. Klart at

$$N = N_1 + N_2 + \cdots + N_{n-1}$$

der N_k er antall veier av lengde k mellom v_1 og v_n. (Det kan jo ikke finnes veier av lengde n eller mer, for da ville veien inneholde samme node flere ganger). Vi har $N_1 = 1$. Videre er $N_2 = n-2$ fordi en vei med to kanter fremkommer ved å velge ut hvilken av de $n-2$ nodene $v_2, v_3, \ldots, v_{n-1}$ som skal være den indre noden i veien. For å finne N_3 skal vi først velge én av nodene $v_2, v_3, \ldots, v_{n-1}$ som første indre node på veien og deretter én av de gjenværende nodene som den andre indre noden. Dette kan gjøres på $(n-2)(n-3)$ ulike måter, så $N_3 = (n-2)(n-3)$. På samme måte finner vi at

$$N_k = (n-2)(n-3)(n-4)\cdots(n-k) \quad (2 \leq k \leq n-1).$$

Så spesielt er $N_{n-1} = (n-2) \times (n-3) \times \cdots \times 2 \times 1 = (n-2)!$. Dette betyr at N, betraktet som en funksjon av antall noder n, vokser raskere enn $(n-2)!$. F.eks. hvis $n = 20$ er N_{19} omtrent 6×10^{15}, og hvis $n = 40$ er N_{39} omtrent 5×10^{44}. Dette er mange veier! Antall veier vokser eksponentielt i n (dvs. som funksjon av n). □

Vi har nå sett at antall veier mellom et par av noder kan være gigantisk hvis grafen er "tett nok". Dette har konsekvenser for optimeringsproblemer. Betrakt igjen *korteste vei problemet* i en graf $G = (V, E)$. Da har vi gitt et ikkenegativt tall l_e for hver kant $e \in E$, og dette tallet kan vi tenke på som lengden av kanten. I vårt eksempel med et veikart er l_e lengden av den gaten som svarer til kanten e. Videre har vi gitt to noder s og t og ønsker å finne en kortest mulig vei G mellom s og t. Med lengden av veien menes summen av lengden til alle kantene som inngår i veien. Den direkte måten å løse dette problemet på er å bestemme alle veiene,

beregne lengdene og finne én med kortest lengde. Men, som eksemplet over viser, vil en slik fremgangsmåte være umulig å bruke fordi antall veier er altfor høyt. Betyr dette at man ikke kan løse korteste vei problemet effektivt? Nei, men man må gå fram på en smartere måte, som vi vil se nærmere på etterhvert.

Subgraf. La G være en graf. Hvis vi fjerner visse noder og/eller kanter i G får vi en *subgraf* av G. Merk: hvis vi fjerner en node, så må vi også fjerne alle kantene som er inntil denne noden, slik at resultatet blir en ny graf! Man bruker gjerne notasjonen $G \setminus S$ for subgrafen der en viss nodemengde S (eller kantmengde, eller begge deler) er fjernet. Hvis S f.eks. bare består av én node v skriver vi gjerne $G \setminus v$ istedet for $G \setminus \{v\}$. F.eks. hvis G er en syklus med noder $v_1, v_2, \ldots, v_7$, så er $G \setminus v_1$ en vei med noder $v_2, v_3, \ldots, v_7$.

1.3 Trær

Vi har sett på begrepet vei i en graf. Nå skal vi se på en mer generell struktur i en graf, nemlig såkalte trær.

Først vil vi definere hva en syklus er. La som vanlig $G = (V, E)$ være en graf. En *syklus* er definert på samme måte som en vei bortsett fra at endenodene nå faller sammen. En syklus av *lengde* k inneholder k kanter og k noder. (Man tenker noen ganger på en vei eller syklus også some en node/kant sekvens, avhengig av hvilken sammenheng begrepet brukes). Vi snakker ofte om en syklus i en graf G, denne er da en syklus der nodene og kantene stammer fra G.

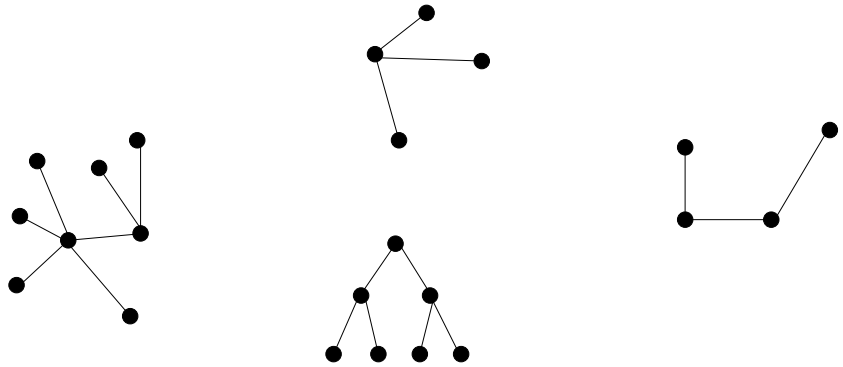
Dette er en grunnleggende egenskap ved en graf. En graf som ikke er sammenhengende er oppdelt i ulike (maksimale) komponenter som hver er sammenhengende. Det er ofte slik for problemer i grafer at man kan nøye seg med å studere problemet for sammenhengende grafer.

Nå er vi kommet fram til definisjonen av et tre. Et *tre* er en sammenhengende graf uten noen syklus. Her er noen eksempler på trær:

Betrakt et tre T og la u og v være to (distinkte) noder i T. Siden T, pr. definisjon, er sammenhengende vil T inneholde en vei mellom u og v. Men kan det være *mer* enn én slik vei? Hvis vi ser på eksemplene over, legger vi merke til at det er nøyaktig én uv-vei. Faktisk gjelder dette generelt.

Teorem 1.2. *La T være et tre og la u og v være to noder i T. Da inneholder T nøyaktig én vei mellom u og v,*

Akkurat som vi har nevnt begrepet "vei i en graf" snakker vi om et tre i en graf. Dette er et tre der nodene og kantene inngår i den gitte grafen. En graf kan



Figur 1.5: Eksempler på trær.

inneholde svært mange trær. Generelt er det vanskelig å finne en pen formel for dette antallet. Derimot kan man spørre om antall distinkte trær med n noder; la oss betegne dette antallet med N_T. Merk at vi her mener *ulike* trær, så mange av disse trærne er isomorfe. Cayley fant på slutten av 1800-tallet ut at

$$N_T = n^{n-2}.$$

Følgende tabell viser dette antallet for noen verdier av n:

n:	3	6	12	20	40
N_T:	3	1296	6×10^{10}	2.6×10^{23}	7.5×10^{60}

For å få bedre innsikt i struturen på trær, skal vi se hvordan disse objektene kan konstrueres. For å illustrere prosedyren, kan vi se på de enkleste trærne, nemlig veier. Enhver vei P med n noder kan konstrueres ved å starte med en vei Q med $n-1$ noder og addere en ny node v og en kant $[v, w]$ der w er én av de to endenodene til veien Q.

Tilsvarende kan vi gå fram for trær. Betrakt følgende prosedyre:

Konstruksjon av trær: *Ethver tre T med n noder kan konstrueres fra et tre T' med $n-1$ noder ved å innføre en ny node og en kant mellom denne noden og én av nodene i T'.*

Hvordan kan vi se at dette er sant? Svaret ligger i neste, ganske enkle, resultat. Med et *ikketrivielt tre* mener vi et tre med minst to noder. Et *blad* i et tre T er en node $v \in T$ med grad 1. Så et blad er inntil nøyaktig én kant i treet. Noen trær har mange blader og noen trær har få! Følgendne resultat er enkelt, men nyttig.

Lemma 1.1. *Ethvert ikketrivielt tre T har minst to blader.*

Bevis: La P være en vei i T med flest mulig noder. Da må P gå mellom to blader i T for ellers kunne T utvides med en node (idet T ikke inneholder noen syklus). □

Korrekthet av trekonstruksjonsprosedyren: La T være et tre med n noder. Hvis $n = 2$ er resultatet klart, så anta $n \geq 3$. La v være et blad i T; dette finnes ved Lemma 1.1. La T' være grafen som fremkommer ved å slette v og dens (unike) nabokant. Vi må bare verifisere at T' er et tre. Klart at T' ikke inneholder noen syklus (for da ville T inneholdt en syklus, som jo er umulig). Videre er T' sammenhengende; dette følger av at de eneste veiene i T som benyttet bladkanten er veiene til bladet v.

Her er et eksempel på et resultat vi får ut fra trekonstruksjonsprosedyren. Det sier oss bl.a. hvor mange kanter det er i et tre.

Teorem 1.3. *La G være en graf med n noder og m kanter. Hvis G er sammenhengende, så må $m \geq n - 1$. Videre har vi likhet her ($m = n - 1$) hvis og bare hvis G er et tre.*

Bevis: Vi ser først på en del av det siste utsagnet. Betrakt trekonstruksjonsprosedyren: et tre med én node har ingen kanter, og hver gang vi utvider treet med en ny node og en ny kant får vi stadig at (antall kanter) = (antall noder) -1. Dette viser at et tre med n noder har $n - 1$ kanter.

La så G være en vilkårlig sammenhengende graf med n noder og m kanter. Vi påstår at G må inneholde et tre med n noder. Fordi: vi kan bygge opp et tre ved å starte med én node og suksessivt føye til en kant til en ny node. Siden G er sammenhengende vil denne prosessen fortsette inntil vi får et tre med alle nodene i G (et slikt tre kalles et spenntre). Men som vist over har dette treet $n - 1$ kanter som dermed viser at enhver sammenhengende graf med n noder må ha minst $n - 1$ kanter. Det følger også fra dette at hvis G har $n - 1$ kanter (og er sammenhengende) så vil G være et tre. □

Trær er også knyttet til begrepet "bro". La G være en sammenhengende graf. Vi kaller en kant $e \in G$ for en *bro* dersom $G \setminus e$ er usammenhengende. Hva kan man si om broer i trær? Jo, vi har følgende resultat.

Teorem 1.4. *La G være en sammenhengende graf. Da er G er tre hvis og bare hvis hver kant i G er en bro.*

Kombinatorisk optimering - hva er det? Ja, dette er en relativt ung gren innen anvendt matematikk der man studerer optimeringsproblemer av kombinatorisk art. Som oftest er disse problemene formulert i grafer. Slike problemer har

mange anvendelser på svært ulike felt innen teknologi, økonomi, planlegging osv. Vi skal se på flere eksempler etterhvert.

1.4 Oppgaver

1. La $A = \{1, 2, \ldots, 50\}$ og la funksjonen $f : A \to \Re$ være gitt ved $f(x) = \sin(2 * \log(\lfloor cos(2x^4 - 5x^3 - \tan(x))\rfloor))$. Hvordan vil du finne minimum av f over A (hvis du disponerer en datamaskin)?

2. Betrakt optimeringsproblemet

$$\begin{array}{lrcll} \text{maksimer} & 3x & + & 2y & \\ \text{forutsatt at} & & & & \\ & x & + & y & \leq 6 \\ & x & & & \leq 4 \\ & & & y & \leq 4 \\ & x & & & \geq 0 \\ & & & y & \geq 0. \end{array}$$

Dette er et LP problem. Illustrer mengden A av tillatte løsninger i planet. Betrakt de punkter $R(\alpha)$ der funksjone $3x + 2y$ er konstant lik α. Hva slags mengde er $R(\alpha)$? Lø LP problemet geometrisk ved å finne i figuren den største verdien av α slik at $R(\alpha)$ snitter den tillatte mengden A. Til slutt et "lite" tilleggsspørsmål: hvordan løser vi LP problemer mer 10 000 variable og 40 000 ulikheter eller likninger? Slike problemer møter man i praksis!

3. K_n har $n(n-1)/2$ kanter. Hvorfor?

4. Tegn K_1, K_2 og K_3 i planet. Kan du tegne K_4 i planet uten kryssende linjer (dvs. linjene som svarer til kantene skal ikke krysse hverandre)? Hva med K_5. Og K_6?

5. Hva er graden til hver node i K_n? Finn en graf med 6 noder der hver node har grad 2! Fins det flere slike? Finn en graf der alle noder har grad 1. Finn en graf med 5 noder der to av dem har grad 2 og de andre har grad 1.

6. På et selskap håndhilser ulike personer på hverandre. Ved slutten av selskapet vil det være et partall antall gjester som har hilset på et odde antall personer. Forklar hvorfor!

7. Betrakt to isomorfe grafen G og G'. Hva kan du si om de tilhørende gradsekvensene? Fins det "en omvending"?

8. Gi et bevis for Teorem 1.2. (Anta at det er to veier og se hva det må bety!)

9. Gi et bevis for Teorem 1.4. (Hint: bruk Teorem 1.3.)

10. La T være et tre, og la u og v være to noder i T som ikke er naboer i T. Hvis vi legger kanten $e = [u, v]$ til T får vi en graf med nøyaktig én syklus. Forklar hvorfor! Hva skjer hvis vi fjerner en annen kant enn e fra denne syklusen, hva slags graf får vi da?

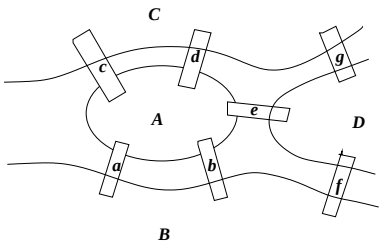
Kapittel 2

Königsberg, Euler, spenntrær og farver!

Vi har nå kommet godt igang med litt grafteori. I dette andre kapitlet vil vi først gå tilbake til grafteorien begynnelse; til byen Königsberg i Øst-Preussen i 1736. Der var man opptatt av et interessant problem som den kjente matematikeren Leonard Euler tok fatt i. Knyttet til en generalisering av dette problemet har vi idag begrepet Euler-tur i en graf. Deretter skal vi studere et viktig optimeringsproblem for trær, det såkalte minimum spenntre problemet. Til slutt ser vi på farvelegging i grafer.

2.1 Starten: Königsberg og Euler

La oss vende blikket mot byen Königsberg i Øst-Preussen på 1700-tallet. Byen var (og er) delt av elven Pregolja i fire landområder som var forbundet med syv broer. En skjematisk tegning av byen slik den var på den tider er som følger:

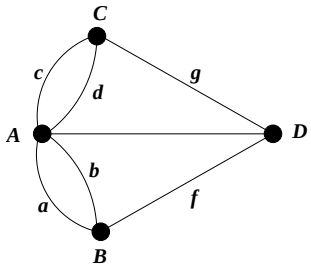


Figur 2.1: Königsberg.

Det sies at man (i visse kretser) var opptatt av følgende problem: er det mulig å finne en rundtur i Königsberg som starter og slutter samme sted og som krysser hver bro nøyaktig én gang? Til tross for mange forsøk var ingen i stand til å finne

en slik rundtur. Så man trodde etterhvert at problemet ikke hadde noen løsning, men ingen var istand til å verifisere dette heller.

Den berømte sveitsiske matematikeren Leonard Euler oppholdt seg i Königs-berg og ble kjent med broproblemet. Han fant snart ut at problemet ikke hadde noen løsning, og ga et bevis for dette. Dette ble beskrevet i artikkelen [9] om ble publisert (på latin) i 1736. Euler lagde her følgende diagram



Figur 2.2: Eulers diagram.

som muligens kan være verdens første graf?

Det viktige med dette arbeidet var at Euler introduserte grafbegrepet. Dette skjedde i tilknytning til broproblemet og ga ikke bare opphav til en analyse av dette lille problemet, men gjorde det mulig å formulere og løse "broproblemet" i generelle grafer. Vi ser på dette i moderne språkdrakt.

En *vandring* i en graf $G = (V, E)$ er en sekvens

$$v_1, e_1, v_2, e_2, v_3, \ldots, v_t, e_t, v_{t+1}$$

der hver v_i er en node, og $e_i = [v_i, v_{i+1}] \in E$ for $i = 1, 2, \ldots, t$. Her er det tillatt at noder og/eller kanter forekommer flere ganger. Vandringen kalles *lukket* dersom $v_1 = v_{t+1}$. Vi kan nå definere det begrepet som svarer til rundturen i Königsberg. En *Euler-tur* er en lukket vandring som inneholder alle kantene i grafen. Broproblemet var derfor å finne en Euler-tur i grafen i Figur 2.2.

Så spørsmålet er: når har en graf en Euler-tur? Svaret ser ved å observere at underveis i en vandring, når vi kommer til en node v, så vil vi bruke nøyaktig *to* av nabokantene til v. Og siden alle kantene skal brukes nøyaktig én gang må hver node ha et partall antall nabokanter for at en Euler-tur skal eksistere. Videre må jo grafen være sammenhengende. Omvendt, hvis disse to betingelsene er oppfylt, så kan man finne en Euler-tur ved å starte i en vilkårlig node v_1 og gå til en nabonode v_2, og deretter til en nabonode v_3 av v_2, osv. Vi vil da hele tiden bruke to kanter inntil hver node og man kan se at dette gir en Euler-tur. Dette

viser Euler's teorem om Euler-turer (faktisk viste Euler bare nødvendigheten av betingelsen; han mente muligens at tilstrekkeligheten var åpenbar).

Teorem 1736. *En graf $G = (V, E)$ har en Euler-tur hvis og bare hvis G er sammenhengende og graden til hver node er et partall.*

2.2 Konnektor og spenntrær

Tenk deg at du får i oppgave å utforme et telekommunikasjonsnettverk der visse punkter skal knyttes sammen med kabler til lavest mulig kostnad. Vi antar at kostanden ved å etablere en direkte forbindelse mellom to punkter i og j er kjent, og vi lar c_{ij} betegne denne kostanden. Vi tenker oss en enkel situasjon der man kan se bort fra kapasiteter (f.eks. er forbindelsene fiberoptiske kabler). Hvilke forbindelser (linker) skal etableres? La oss kalle dette problemet *minimum konnektor problemet.*.

Vi kan angripe minimum konnektor problemet på følgende vis. Først innfører vi en graf $G = (V, E)$ med én node for hvert av de gitte punktene som skal knyttes sammen. Videre lar vi det være en kant mellom node i og node j dersom det er mulig å etablere en direkte forbindelse mellom de tilhørende punktene (det er kjent om dette er mulig), og knyttet til denne kanten har vi kostnadstallet c_{ij}. Minimum konnektor problemet er da å finne en delmengde F av kantmengden E slik at

1. subgrafen (V, F) er sammenhengende, og
2. $\sum_{ij \in F} c_{ij}$ er minst mulig.

Her er altså punkt 1 et krav til valg av F mens punkt 2 uttrykker målet: å finne en F med lavest vekt blant alle kantdelmengde som gir en sammenhengende subgraf. For at problemet skal ha mening (og tillatte løsninger finnes) antar vi at G er sammenhengende.

Vi vil først si noe om *strukturen* på optimale løsninger. Det er greit å definere kostnaden til en kantdelmengde $F \subseteq E$ ved

$$c(F) := \sum_{e \in F} c_e.$$

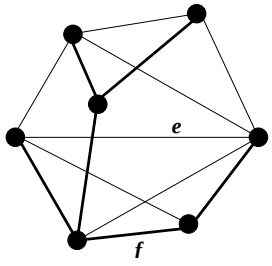
Så dette er summen av kostnadene til alle kantene i F.

Anta først at alle kostnadene er positive (negative kostnader er unaturlige i dette problemet). Vi definerte tidligere en *bro* i en sammenhengende graf H som en kant som er slik at $H \setminus e$ er usammenhengende.

Betrakt nå grafen $G = (V, E)$ i minimum konnektor problemet. Siden G er sammenhengende er opplagt E en tillatt løsning ("tillatt" i betydningen at (V, E) er sammenhengende) og den tilhørende kostnaden er $c(E)$ (summen av alle kostnadene). Vi vil gjerne redusere kostnaden ved å fjerne kanter; kantene har jo positive kostnader. Så velg en kant $e \in E$. Kan e fjernes? Hvis e er en bro, så er $G \setminus e$ ikke sammenhengende og da vil ikke $E \setminus \{e\}$ være en tillatt løsning. Men hvis derimot e ikke er en bro, kan e fjernes og vi får en ny tillatt løsning som har lavere kostnad. Dette betyr at vi kan suksessivt lete etter en kant som ikke er en bro (i den subgrafen vi betrakter) og fjerne denne kanten. Hvor langt kan denne reduksjonen føre oss? Jo, inntil vi har en subgraf (V, F) som er sammenhengende og bare inneholder broer. Men ved Teorem 1.4 innebærer dette at (V, F) er et tre.

Altså: vi har sett at enhver optimal løsning av minimum konnektor problemet i grafen $G = (V, E)$ er et tre (V, F). Et slikt tre, dvs. et tre i G som inneholder alle nodene i G, kalles et *spenntree* (idet det utspenner alle nodene i G). Et eksempel på] en graf og et spenntre (uthevede linjer) i grafen er gitt i Figur Dette betyr at sålenge kostnadene er positive vil minimum konnektor problemet redusere seg til følgende problem.

Minimum spenntre problemet: *I en gitt graf med ikkenegative kostnader knyttet til kantene, finn et spenntre med minimum total kostnad.*



Figur 2.3: Spenntre i en graf.

Så derfor skal se nå se på hvordan vi kan løse minimum spenntre problemet. Men først et naturlig spørsmål!

Minimum spenntre problemet er et eksempel på et kombinatorisk optimeringsproblem. Det er en kommentar om slike problemer som vi bør komme med. Felles for nær sagt alle kombinatoriske optimeringsproblemer er at man skal finne en "optimal delmengde" av en viss endelig grunnmengde. Det er da bare visse delmengde som er tillatte. For eksempel, i minimum spenntre problemet er bare spenntrær tillatte, og grunnmengden er rett og slett kantmengden i den gitte

grafen. Som oftest i slike problemer vil antall tillatte delmengder bli gigantisk når størrelsen på grunnmengden øker. Som illustrasjon, hvis G er den komplette grafen med n noder, er antall kanter $m = n(n-1)/2$ og antall spenntrær er m^{m-2} som er av orden $(n^2)^{n^2}$. Dette betyr at, unntatt hvis n er svært liten, kan problemet ikke løses ved den opplagte metoden: finn alle de tillatte løsningene, sammenlikne kostanadene og velg en løsning med lavest kostnad. Vi trenger derimot en algoritme der regnetiden ikke vokser for fort i forhold til størrelsen på grunnmengden. En slik algoritme finnes her!

Før vi ser på en slik algoritme for minimum spenntre problemet skal vi gi en lite, men nyttig resultat om spenntrær, se igjen Figur 2.3.

Proposisjon 2.1. *La T være et spenntre i en sammenhengende graf $G = (V, E)$ og la e være en kant i G som ikke ligger i T. Da fins det en kant f i T slik at hvis man erstatter f med e i T så får man et nytt spenntre i G.*

Vi kan nå gi en enkel og effektiv algoritme for minimum spenntre problemet. Enkelheten ligger i at man gjør det som er mest fristende, nemlig å bruke de kantene som har lavest kostnad først.

Kruskal's algortme for minimum spenntre problemet:

1. Sorter de m kantene med ikkeavtagende kostnader, si at $c_{e_1} \leq c_{e_2} \leq \cdots \leq c_{e_m}$.
2 i. Start med $F = \emptyset$.
2 ii. Sålenge $|F| < n-1$, utvid F med den neste kanten i sekvensen over som er slik at (V, F) ikke inneholder noen syklus.

Når algoritmen terminerer vil F inneholde $n-1$ kanter og disse utgjør et spenntre, faktisk et spenntre med lavest mulig kostnad! Algoritmen velger altså ut kantene med lavest kostnad på en slik måte at den hopper over de kantene underveis som ville gitt opphav til en syklus. Dette er en effektiv algoritme. Hoved-arbeidet er faktisk det som skjer i skritt 1 der man skal sortere i m kantene etter deres kostnader. Til dette finnes ulike sorteringsalgoritmer (quick-sort, boble-sort osv), og de raskeste algoritmene krever orden $n \log n$ aritmetiske operasjoner (subtraksjoner). Siden logaritmefunksjonen vokser svært langsomt, er dette for praktiske formål nærmest å regne for en lineær funksjon av m.

I skritt 2 har vi høyst m iterasjoner (selv om vi vet at $n-1$ kanter skal finnes). I hver iterasjon skal man avgjøre om grafen (V,$F \cup \{e\}$) inneholder noen syklus. Dette kan gjøres ved at man merker nodene i treet indusert av F. En ny kant kan da legges til F hvis minst én av endenodene er umerket (for hvis begge er merket ville man få et tre med en kant mellom to noder; dette gir en syklus. En slik

merking implementeres ved å bruke en array av lengde n og med f.eks. verdier 0 og 1 (1 kan bety "merket"). Dette innebærer at det å teste om en kant kan legges til treet gjøres på en effektiv måte.

Algoritmen over ble funnet av Kruskal på slutten av 1950-tallet og den kalles en *grådig-algoritme*. Navnet indikerer at man er grådig i hvert skritt: tar det som da er best såfremt det kan komme til nytte! Det er derfor hyggelig at denne grådigheten virker for dette problemet. Faktisk vil algoritmen virke for vilkårlige kostnader, uansett om noen skulle være negative eller null. Vi snakker derfor om "vekter" knyttet til kantene (vilkårlige tall).

Teorem 2.1. *La G være en sammenhengende graf med vekter knyttet til kantene. Da vil Kruskals algoritme finne et spenntre med minimum vekt.*

Bevis: Det er klart at algoritmen finner et spenntre, så vi kan konsentrere oss om å vise optimaliteten. La T være spenntreet algoritmen finner, og la T^* være et minimum vekt spenntre som har flest mulig kanter til felles med T. Det er nok å vise at $T = T^*$. Anta istedet at $T \neq T^*$ og la e_k være den første kanten (i kantsekvensen etter ordningen) som ligger i T men ikke i T^*. Dette betyr at kantene $e_1, e_2, \ldots, e_{k-1} \in T \cap T^*$. Ved Proposisjon 2.1 fins det en kant f i T^* slik at $T^{**} = (T^* \setminus \{f\}) \cup \{e_k\}$ er et spenntre i G. Hva kan vi si ellers om T^{**}? Siden T^* er et spenntre av minimum vekt må $c(T^*) \leq c(T^{**})$ som gir at

$$c_f \leq c_{e_k}.$$

På den annen side kan ikke kantene $e_1, e_2, \ldots, e_{k-1}, f$ inneholde en syklus (fordi alle disse kantene ligger i spenntreet T^*). Men dette betyr at da algoritmen (som ga T) valgte kanten e_k var også f en kandidat, så følgelig må

$$c_{e_k} \leq c_f.$$

Dermed har vi vist at $c_{e_k} = c_f$. Konsekvensen av dette er at T^* og T^{**} har samme vekt. Følgelig er også T^{**} et minimum vekt spenntre. Videre er

$$|T^{**} \cap T| > |T^* \cap T|$$

og dette strider mot at T^* var et minimum vekt spenntre med flest mulig kanter felles med T. Denne motsidelsen viser at $T = T^*$ og derfor er T en optimal løsning. □

2.3 Farvelegging

I denne korte seksjonen vil vi kort omtale farvelegging i grafer.

Tenk deg et kart over Europa der (som regel) de ulike statene skilles fra hverandre ved ulike farver. Dette er gjort slik at land med felles grense ikke har samme farve (bortsett fra hvis landene bare møtes i ett punkt, da aksepteres samme farve). Spørsmålet er: hvor få farver kan man klare seg med? Dette spørsmålet ble beskrevet og studert en del på midten av 1800-tallet i England. Man var da opptatt av dette problemet for enhver tenkelig konfigurasjon av stater (ikke bare Europa). En student ved navn Frederick Guthrie stilte spørsmålet til de Morgan (professor i matematikk ved University College i London) som igjen diskuterte problemet i brev til Sir William Hamilton. Dermed var snøballen igang og ulike matematikere og andre ble opptatt av problemet.

Problemet hadde nok vært kjent en stund innen kartkretser, og man trodde at det alltid var mulig å klare seg med 4 farver, men ingen kunne si dette med sikkerhet.

Men i 1879 skjedde en stor begivenhet. Alfred Kempe (annonserte først og) publiserte et bevis for *Fire-farve teoremet* i tidsskriftet *American Journal of Mathematics*. Denne artikkelen er gjengitt i boken [4], som gir en fantastisk fin historisk framstilling av grafteori deriblant farvelegging. Kempe's bevis vakte stor oppsikt og anerkjennelse.

I 1890 ble det funnet en feil i Kempe's bevis! Og Fire-farve teoremet skiftet navn til "Fire-farve teoremet". Percy J. Heawood publiserte en artikkel der han påviste en feil i Kempe's argument. Dette vakte en viss skuffelse i de matematiske kretser og problemet var igjen uløst. Men Heawood gjorde mer enn å "ødelegge", han viste "Fem-farve teoremet", at ethvert kart kan farvelegges med høyst 5 farver. Faktisk lå denne idéen i Kempe's arbeid.

Først i 1976 ble saken avklart da K. Appel og W. Haken publiserte et bevis for Fire-farve teoremet som ble akseptert[1]. Imidletid er beviset utradisjonelt ved at man brukte et data program til å sjekke ut en lang rekke tilfeller i en viss dislkusjon. Senere har beviset blitt noe forenklet, men stadig med svært mange konfigurasjoner (ca. 700) som må betraktes.

La oss se hvordan farveleggingsproblemet kan knyttes til grafer. For et gitt kart lager vi en graf med en node for hvert land og en kant mellom naboland (som møtes langs en grense bestående av mer enn ett punkt). En slik graf er *planar*. Dette innebærer at grafen kan tegnes i planet uten kryssende linjer (der linjer svarer til kanter).

1. Så nå er det ikke lenger "Fire-farve teoremet"!

La $G = (V, E)$ være en vilkårlig graf. En *farvelegging* av grafen er en tilordning av en farve til hver node slik at nabonoder har ulik farve. Mer formelt er en farvelegging en funksjon $c : V \to \{1, 2, \ldots\}$ slik at $c_u \neq c_v$ når $[u, v] \in E$. Så tall svarer til farver, og vi kan anta at de tallene som brukes er $1, 2, 3, \ldots$ (dvs. vi hopper ikke over noen tall). *Farvetallet* til G er det laveste antall farver i en farvelegging av G, og dette antallet betegnes med $\chi(G)$. Fire-farve teoremet sier dermed at $\chi(G) \leq 4$ for enhver planar graf. Vi kan forresten nevne et interessant teorem av Grötzsch (fra 1959) som sier at $\chi(G) \leq 3$ for enhver planar graf som ikke inneholder noen triangler (et triangel er tre noder der hvert par er naboer).

Appel og Haken viste altså Fire-farve teoremet.

Teorem 2.2. *Enhver planar graf har en farvelegging med høyst fire farver.*

Vi bemerker at Heawood's bevis for Fem-farve teoremet ikke er så langt. Det benytter Euler's polyederformel og gir opphav til en algoritme for å finne en slik farvelegging.

Ellers er farvelegging av *vilkårlige* grafer et vanskelig problem: dette er et *NP-hardt* problem. I praksis betyr dette at ingen effektiv algoritme for problemet er kjent, så man må nøye seg med heuristikker som er raske men ikke kan garantere optimalitet.

En slik enkel, men mye brukt, heurestikk er følgende.

Sekvensiell farvelegging.

1. Ordne nodene slik at gradene er ikkevoksende. Gi første node farve 1.
2. For $k = 2, 3, \ldots, n$ gi node k den laveste mulige farve som er ulik alle farvene som naboene så langt har fått.

Av denne algoritmen finner vi en enkel øvre skranke på farvetallet til en vilkårlig graf. La $\Delta(G)$ betegne den største graden til en node i en graf G.

Proposisjon 2.2. *For enhver graf G er*

$$\chi(G) \leq \Delta(G) + 1.$$

2.4 Oppgaver

1. Betrakt en graf G. Definer en *Euler-vandring* i G som en ikke-lukket vandring som inneholder alle kantene i grafen . Når vil en graf inneholde en Euler-vandring?

2. Betrakt forrige problem. Ser du noen sammenheng mellom Euler-vandring og problemet å finne veien ut av en labyrint (med én utgang, og du er plassert et sted inne i labyrinten).

3. Tegn en graf med 20 noder som oppfyller betingelsene i Teorem 1736. Finn en Euler-tur!

4. Hvordan kan vi løse minimum konnektorproblemet når enkelte kanter har "kostnad" som er ikkepositive?

5. Finn et minimum spenntre i grafen gitt i Figur 2.4.

6. Vis Proposisjon 2.1.

7. La G være en graf der hver kant er farvet rød eller blå. Forklar hvordan man kan finne et spenntre med flest mulige røde kanter!

8. Vi sier at to spenntrær (i en gitt graf) er *naboer* dersom de er like bortsett fra for én kant. Vis at for ethvert par T, T' av spenntrær så finnes det en sekvens av spenntrær $T_1, T_2, \ldots, T_k$ der $T_1 = T$, $T_k = T'$ og der T_i og T_{i+1} er naboer for $i = 1, 2, \ldots, k-1$.

9. Anta at en sammenhengende graf G har positive kantvekter. Betrakt følgende algoritme for minimum konnektorproblemet. Ordne kantene med ikkevoksende vekter. Start med alle kantene og fjern så kanter fra starten av sekvensen men behold hver gang kanten hvis den gjenværende grafen ville bli usammenhengende. Forklar hvorfor denne algoritmen løser minimum konnektor problemet. Hva med minimum spenntre problemet?

10. Anta at G er en sammenhengende graf med ikkenegative kantvekter. La videre F være en gitt delmengde av kantmengden. Hvordan kan man finne et spenntre T av minimum vekt der vi krever at T skal inneholde F?

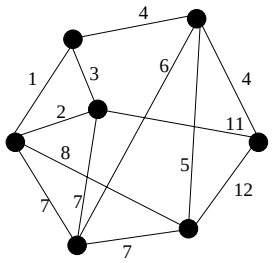
11. ta for deg et kart over f.eks Europa og finn en farvelegging. Hvor mange farver brukte du? Kan du eventuelt vise at dette er en optimal farvelegging?

12. Forklar hvorfor Proposition 2.2 må holde.

13. Hvorfor ordnes nodene etter avtagende grad i Sekvensiell farvelegging?

14. Tegn en graf med f.eks. 20 noder. Bruk algorithen Sekvensiell farvelegging. Prov noen forskjellig ordninger av nodene før du gjennomforer steg 2 i algoritmen. Hva skjer?

15. Vis at med passende ordning av nodene vil steg 2 i Sekvensiell farvelegging finne en optimal farvelegging.



Figur 2.4: Spenntreoppgave.

Kapittel 3

Veier i grafer og strøm i nettverk

Vi nevnte tidligere problemet å finne en korteste vei mellom to punkter i et veinettverk. Dette er et eksempel på *korteste vei problemet* i en vektet graf som vi skal studere i dette kapitlet. Vi skal også se nærmere på et par andre optimeringsproblemer i nettverk: *maksimum strøm problemet* og *minimum kutt problemet*.

3.1 Korteste vei problemet

I denne seksjonen ser vi på en *vektet graf*, dvs. en graf med et ikkenegativt tall knyttet til hver kant. I anvendelser vil dette tallet kunne representere lengde, avstand, kostnader, inntekter eller liknende. Vi er opptatt av problemet å finne en kortest mulig vei mellom to gitte noder i grafen.

La $G = (V, E)$ være en vektet graf med vektfunksjon l gitt ved l_e for hver $e \in E$. *Lengden* (eller *vekten*) $l(P)$ til en vei P defineres ved

$$l(P) = \sum_{e \in P} l_e$$

der summen tas over alle kanter i veien. (Merk: en vei kan oppfattes som en kantsekvens, en nodesekvense eller en node/kant-sekvens, helt avhengig av sammenhengen).

Vi kan derfor snakke om avstander i en sammenhengende graf G. Definer $d(u, v)$ som lengden av en korteste vei mellom node u og node v i G; vi kaller denne størrelsen for *avstanden mellom u og v*. Vi antar her at de underliggende kantvektene l_e er gitt og holdes fast. Hvis alle kantvektene er positive kan man vise at denne funksjonen $d : V \times V \to \Re_+$ definerer en *metrikk* på V, og (V, d) er et (endelig) metrisk rom. Dette innebærer at følgende egenskaper holder for alle $x, y, z \in V$:

(i) $d(x, y) = 0$ hvis og bare hvis $x = y$,
(ii) $d(x, y) = d(y, x)$, og
(iii) $d(x, z) \leq d(x, y) + d(y, z)$ (trekantulikheten).

Figur 3.1: Eksempler vektet graf og tilhørende metrikk.

Hvis $l_e \geq 0$ får vi en såkalt *pseudometrikk* der (i) ikke nødvendigvis holder. Et eksempel på en (enkel) vektfunksjon er $l_e = 1$ for alle kanter e. Da er $d(u, v)$ lik det laveste antall kanter i en uv-vei i G. Her er problemet vi fokuserer på.

Korteste vei problemet: *Gitt en vektet graf G med vektfunksjon l og to noder r og t, finn en rt-vei P med minimum lengde, dvs. P oppfyller $l(P) = d(r, t)$.*

Vi antar altså at vektene er ikkenegative. Det er slik at problemet faktisk kan være ganske annerledes hvis man tillater negative vekter (beregningsmessige vanskelige problemer blir da et spesialtilfelle).

Det å finne en god algoritme for et matematisk problem henger nøye sammen med det å få en god forståelse for strukturen i det aktuelle problemet. For korteste vei problemet ligger "hemmeligheten" i følgende enkle resultat om avstander.

Lemma 3.1. *Definer $d_v = d(r, v)$ som avstanden fra r til node v (med vektfunksjon l). Da gjelder*

$$d_v \leq d_u + l_{uv} \quad \text{for alle } [u, v] \in E.$$

Videre, hvis $P : r = v_0, v_1, \ldots, v_k = t$ er en rt-vei som oppfyller at

$$d_{v_{i+1}} = d_{v_i} + l_{v_i v_{i+1}} \quad \text{for } i = 0, 1, \ldots, t - 1$$

så er P en korteste rt-vei i G.

Bevis: Anta at $[u, v] \in E$. Det fins en ru-vei P' med lengde $d_u = d(r, u)$. Hvis P inneholder node v er jo $d_v \leq d_u \leq d_u + l_{uv}$ idet kantvektene er ikkenegative. Hvis P ikke inneholder v er P pluss kanten $[u, v]$ en rv-vei med lengde $d_u + l_{uv}$ så også da er $d_v \leq d_u + l_{uv}$.

Den siste delen av lemmaet følger direkte av antagelsen idet lengden av P er summen av kantlengdene

$$l_{v_i v_{i+1}} = d_{v_{i+1}} - d_{v_i}$$

som blir $d_t - d_r = d_t = d(r,t)$. □

En annen måte å uttrykke dette resultatet på er: *hvis P er en korteste vei fra r til t, så må også delveien av P fra r og fram til en indre node v være en korteste rv-vei i G.*

Hvordan kan vi konstruere en algoritme for korteste vei problemet ut fra Lemma 3.1? La oss tenke oss at vi har allerede har funnet en korteste vei fra r og til visse noder, si alle noder i en mengde S. Da kjenner vi for hver node $u \in S$ både avstanden $d_u = d(r,u)$ og en korteste vei P_u fra r til u. Vi kan her anta at $r \in S$ fordi $d_r = d(r,r) = 0$ og den trivielle veien med bare noden r er kortest. Videre kan vi anta at alle disse korteste veiene P_u ($u \in S$) bare består av noder i S. Grunnen er (som vi fant over) at når vi har funnet en korteste vei P_u til u så har vi også funnet en korteste vei til hver node i P_u, nemlig delveien fram til denne noden. Derfor kan vi like godt anta at alle disse nodene ligger i S.

Betrakt så en node v som ikke ligger i S. Vi vil gjerne finne en korteste vei til denne noden. For hvis vi klarer dette, kan vi jo bare utvide S med node v. Det viser seg nå at hvis vi velger denne noden v på en passende (smart!) måte så blir det enkelt å finne en korteste vei dit!

Lemma 3.2. *Betrakt situasjonen over og definer for hver $u \in S$ og $v \notin S$*

$$P^+(u,v) = P_u \cup \{[u,v]\}$$

dvs. vi tar den korteste ru-veien P_u og føyer til kanten $[u,v]$. Velg nå $u \in S$ og $v \notin S$ slik at

$$l(P^+(u,v)) = d_u + l_{uv}$$

er minst mulig. Da er $P^+(u,v)$ en korteste rv-vei og

$$d_v := d(r,v) = d_u + l_{uv}. \tag{3.1}$$

Bevis: Enhver vei Q fra r til en node $w \notin S$ må starte med en vei i S pluss en kant fra en node i S til en node i $V \setminus S$. Og denne delen av veien har lengde minst lik minimumet i lemmaet. Videre vil denne nedre skranken oppnås nettopp ved det valget som beskrives i lemmaet. □

Så ved dette lemmaet ser vi hvordan vi finner en node v som vi utvider S med. I starten lar vi $S = \{r\}$ og deretter kommer flere iterasjoner der hver iterasjon består i å finne en nye node v som vi utvider S med. Da får vi en ny nodemengde S med de samme egenskapene: vi kjenner alle avstandene fra r til nodene i S og også tilhørende korteste veier som dessuten ligger i S. Beregningene er, som vi ser fra lemmaet, knyttet til bestemmelsen av minimumet av $d_u + l_{uv}$ over alle kanter $[u, v]$ der $u \in S$ og $v \notin S$. Det blir høyst $n - 1$ slike iterasjoner, kanskje mye færre hvis vi er heldige å treffe på t i en tidlig iterasjon.

Denne algoritmen løser korteste vei problemet. Den ble funnet av Dijkstra på slutten av 1950-tallet. Det er et interessant trekk ved denne algoritmen at den faktisk finner en korteste vei fra r og til enhver annen node, ikke bare t! Denne ytterligere informajonen får vi altså gratis på kjøpet, og dette er nyttig i en rekke anvendelser.

Vi gir nå Dijkstra's algoritme. Den er som beskrevet over, men i tillegg brukes et merke $d(v)$ for hver node v. $d(v)$ representer lengden av en viss rv-vei og oppdateres etterhvert som kortere veier finnes. Vi har at $d(v) \geq d_v = d(r, v)$ underveis, og når algoritmen terminerer er $d(v) = d_v$ for alle $v \in V$.

Dijkstra's korteste vei algoritme.

0. La $S := \{r\}$, $d(r) := 0$, $d(v) = \infty$ for hver $v \in V \setminus \{r\}$.
   Hvis $|V| = 1$, stop.
1. For hver $v \notin S$ gjør følgende:
   hvis minimum av $d(u) + l_{u,v}$ for $u \in S$ og $[u, v] \in E$ er mindre enn $d(v)$, så oppdateres $d(v)$ til dette minimumet og la node v få nytt merke lik $(d(v), u)$.
2. Bestem $\min_{v \in \bar{S}} d(v)$ og la v være en node der dette minimumet oppnås.
3. Oppdater S ved $S := S \cup \{v\}$. Hvis $S = V$, så stopp;
   ellers returner til skritt 2.

Noen sluttkommentarer:

- Dijkstra's algoritme har kompleksitet $O(n^2)$ der n er antall noder i grafen. Dette betyr at antall regneopersjoner vokser som en (konstant ganger) n^2.
- Merkeoppdateringen i steg 2 forekommer når man finner en kant $[u, v]$ der $d(v) > d(u) + l_{uv}$. En slik kant kalles *gal* idet funksjonen $d(\cdot)$ ikke oppfyller Lemma 3.1. Det finnes andre algoritmer for korteste vei problemet som oppdaterer $d(\cdot)$ ved å lete etter kanter som er gale, og så rette disse opp ved å sette $d(v) := d(u) + l_{uv}$. Ved en viss systematikk vil man ende opp med at d faktisk representerer avstandene fra r og samtidig ha korteste

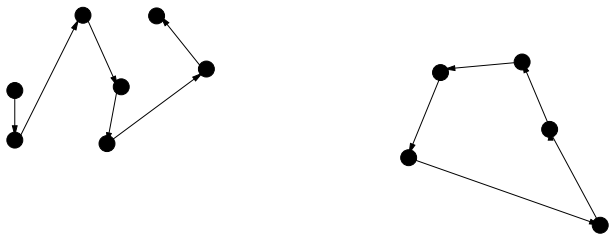
veier fra r og til enhver annen node.

- Korteste vei problemer er aktuelle i langt flere sammenhenger enn for veinettverk. Ofte kan problemer av tilsynelatende helt ulik karakter oversettes til et korteste vei problem i en passende vektet graf. Dette gjelder for eksempel for såkalte Markov beslutingsproblemer (beslutninger tas ved ulike diskrete tidspunkt for et dynamisk system med endelig tilstandsrom).

3.2 Rettede grafer

Det finnes også rettede grafer der kantene har en viss retning. En *rettet graf* er et ordnet par $D = (V, E)$ der V er en endelig mengde av *noder* (eller *punkter*) og E er en endelig mengde av ordnede par av noder. Hvert slikt ordnet par $e = (u, v)$ kalles en *rettet kant* (eller *linje*) og u kalles *startnoden* (eller *initiell endenode* eller *halen*) til e og v kalles *sluttnoden* (eller *terminal endnode* eller *hode*) til e.

Man kan definere begrepene *rettet vei* og *rettet syklus* på samme måte som i en graf, bortsett fra at man nå krever at linjene har samme retning (slik at ingen kanter har felles startnode eller felles sluttnode), Figur 3.2.



Figur 3.2: Rettet vei og rettet syklus.

For hver node $v \in V$ defineres den *utgående stjernen* til v ved

$$\delta^+(v) = \{e \in E : e = (v, w) \text{ for en } w \in V\}.$$

Denne mengden består altså av alle linjer med startnode i v. Tilsvarende består den *inngående stjernen* $\delta^-(v)$ av alle linjer med v som sluttnode. Vi har da at

$$\sum_{v \in V} |\delta^+(v)| = \sum_{v \in V} |\delta^-(v)|.$$

Utgående og inngående stjerner er spesielle kantmengder. En mer generell klasse av delmengder av kantmengden er såkalte "kutt". La S være en ikketom, ekte delmenge av V. Da kalles kantmengden

$$\delta^+(S) := \{(u, v) \in E : u \in S, v \in V \setminus S\}$$

et *rettet kutt* i grafen D. Vi sier også at dette kuttet er *indusert* av nodemengden S. Akkurat som antall veier i en graf gjerne er svært stort, vil antall kutt også være stort. Det er klart av antall nodedelmengder S av V er 2^{n-1}. Hvis grafen er komplett vil også antall rettede kutt være lik 2^{n-1}, mens for ikkekomplette grafer vil ulike delmengder S kunne indusere samme kutt. Likevel vil antall rettede kutt "normalt" være astronomisk stort. Dette spiller en rolle for oss når vi snart skal studere problemet å finne et rettet kutt (av en viss type) som minimerer en viss funksjon.

3.3 Maksimum strøm og minimum kutt

I denne seksjonen ser vi på to ulike problemer under ett. Til tross for at disse problemene er ganske forskjellige er de relatert på en interessant måte. Begge problemer involverer rettede grafer.

Vi skal studere strøm i nettverk. Med et *nettverk* menes rett og slett en rettet graf med kapasiteter knyttet til kantene. La $D = (V, E)$ være en rettet graf og la $c_{u,v}$ være et ikkenegativt tall for hver rettet kant (u, v). Vi oppfatter $c_{i,j}$ som kapasiteten til kanten (i, j) og skal nå definere en strøm i nettverket.

Som motivasjon kan vi tenke oss et veinettverk med biltrafikk med jevn strøm. Da vil den mengden som strømmer inn til et kryss også strømme ut fra krysset (vi ser bort fra bilstopp midt i krysset!). På hver gate er det en kjent kapasitet (bestemt av antall felt og et visst tidsintervall vi ser på). Vi er interessert i hvor mange biler som kan komme gjennom nettverket fra et punkt s til et annet punkt t i løpet av en gitt tidsperiode. Da må vi finne ut hvor mye som skal strømme langs hver gate.

Matematisk vil vi representere en strøm ved en funksjon $x : E \to \Re_+$. Dette betyr at for hver rettet kant $(u, v) \in E$ har vi en variabel $x_{i,j}$ som skal si hvor mye som strømmer i kanten (u, v) fra node u til node v. Videre har vi gitt to noder s og t. I enhver node v ulik s og t krever vi at den totale strømmen inn til v er lik den totale strømmen ut fra v. Matematisk uttrykkes dette slik

$$\sum_{(u,v)\in\delta^-(v)} x_{u,v} = \sum_{(v,w)\in\delta^+(v)} x_{v,w} \text{ for alle } v \in V \setminus \{s, t\}. \tag{3.2}$$

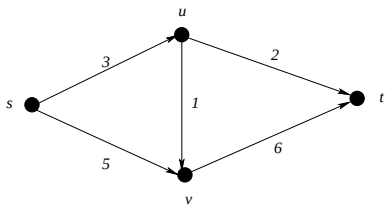
Dette kalles *strømbevaring*. Denne egenskapen er i elektrisitetslære kjent som en av Kirchoffs lover. Videre krever vi at kapasitetene skal overholdes:

$$0 \leq x_{u,v} \leq c_{u,v} \text{ for alle } (u, v) \in E. \tag{3.3}$$

Hvis funksjonen x tilfredsstiller kravene (3.2) og (3.3), så kalles x en *st-strøm*. Til enhver *st*-strøm er det knyttet et tall som kalles strømmens *verdi* og er definert ved

$$v(x) = \sum_{(s,v)\in E} x_{s,v}.$$

Så $v(x)$ er lik mengden som strømmer ut fra noden s. Vi vil heretter anta at D ikke inneholder noen kant med s som sluttnode og heller ingen kant med t som startnode. Man kaller ofte s for *kilden* og t for *sluket*. Det er enkelt å vise at den mengden som strømmer inn til t også er lik $v(x)$. Figur 3.3 viser en *st*-strøm med verdi 8 (vi antar at alle kapasiteten er tilstrekkelig store).



Figur 3.3: En *st*-strøm.

Med denne bakgrunn kan vi definere følgende problem.

Maksimum strøm problemet: *Gitt en rettet graf $D = (V, E)$, to noder s og t, samt en ikkenegativ kapasitetsfunksjon c, finn en st-strøm x med størst mulig verdi $v(x)$.*

Hva slags problem er dette? Dette er et lineær programmeringsproblem. Vi skal maksimere den linære funksjonen $v(x)$ i de reelle variablene $x_{u,v}$ ($(u, v) \in E$) og begrensingene på x er lineære likninger (3.2) og lineære ulikheter (3.3). Dette betyr at man kan bruke generelle LP algoritmer (f.eks. simpleksmetoden) for å løse maksimum strøm problemet. Imidlertid finnes det enklere og mer effektive algoritmer som vi snart skal se noe på. Igjen er poenget å utnytte den spesielle strukturen i problemet. Først skal vi imidlertid slå fast at det virkelig finnes en maksimum strøm.

Lemma 3.3. *Det finnes en maksimum st-strøm, dvs. en st-strøm x med størst mulig verdi $v(x)$.*

Bevis: Verdien til en *st*-strøm x er $v(x) = \sum_{(s,v)\in E} x_{s,v}$ som er en (lineær) kontinuerlig funksjon av x. Videre er mengden av alle *st*-strømmer en lukket og begrenset mengde i vektorrommet $\Re^E$ (faktisk et begrenset polyeder). Ved

Weierstrass' teorem (ekstremverdisetningen) vil da v oppnå sitt maskimum over denne mengde slik at en maksimum strøm eksisterer. □

Vi bemerker at dette resultatet også er en direkte konsekvens av teori for lineær programmering.

Husk at kuttet indusert av en nodemengde S er kantmengden $\delta^+(S)$ som består av alle rettede kanter med startnode i S og sluttnode utenfor S. Vi definerer *kapasiteten til kuttet* ved

$$c(\delta^+(S)) := \sum_{(u,v)\in\delta^+(S)} c_{u,v}$$

som da er summen av kapasitetene til de kantene som inngår i kuttet. Spesielt er vi interessert i *st-kutt* som er rettede kutt $\delta^+(S)$ der S inneholder kildenoden s men ikke sluknoden t. Grunnen er at strømmen fra s til t må passere gjennom ethvert *st*-kutt. Vi kan tenke på et *st*-kutt som en flaskehals i nettverket. Det er da naturlig å spørre etter den laveste kuttkapasiteten fordi, intuitivt, vil dette svare til den "verste flaskehalsen". Betrakt grafen i Figur 3.3 og anta at alle kapasiteter er lik 6. Da er f.eks. $\delta^+(\{s,u\}) = \{(s,v),(u,t),(u,v)\}$ et *st*-kutt, og det har kapasitet 18. Kuttet indusert av $S = \{s\}$ består av kantene (s,u) og (s,v) og har kapasitet 12.

Minimum kutt problemet: *Gitt en rettet graf $D = (V,E)$, to noder s og t, samt en ikkenegativ kapasitetsfunksjon c, finn et st-kutt $\delta^+(S)$ med lavest mulig kapasitet.*

Vi skal nå relatere disse to problemene. For en delmengde F av kantmengden E vil vi heretter benytte notasjonen $x(F) = \sum_{e\in F} x_e$ og tilsvarende $c(F) = \sum_{e\in F} c_e$.

Lemma 3.4. *Hvis x er en st-strøm og $\delta^+(S)$ er et st-kutt, så er verdien til x høyst lik kapasiteten til kuttet, dvs.*

$$v(x) \le c(\delta^+(S)).$$

Bevis: Siden x tilfredstiller strømbalanselikningene for alle noder i $S \setminus \{s\}$ får vi

$$\begin{aligned} v(x) = \; & x(\delta^+(s)) = \textstyle\sum_{v\in S}(x(\delta^+(v)) - x(\delta^-(v))) = \\ & x(\delta^+(S)) - x(\delta^-(S)) \le x(\delta^+(S)) \le c(\delta^+(S)). \end{aligned}$$

De siste to ulikhetene kommer fra kapasitetsbegrensningene (3.3). □

Dette lemmaet har viktige konsekvenser.

Lemma 3.5. *Verdien av maksimum strøm er mindre enn eller lik minimum kutt kapasitet.*

Bevis: Fra Lemma 3.4 har vi at $v(x) \leq c(\delta^+(S))$ for enhver st-strøm x og ethvert st-kutt $\delta^+(S)$. Ved å ta maksimum over alle st-strømmer (og maksimum oppnås ved Lemma 3.3) og deretter minimum over alle st-kutt får vi det ønskede resultatet. □

Faktisk skal vi snart vise at ulikheten i forrige lemma holder med likhet! Dette resultat, som kalles *maks strøm - min kutt teoremet*, regnes som et av de mest sentrale resultatene i kombinatorikk og kombinatorisk optimering. Vi skal vise dette på en konstruktiv måte, dvs. vi vil samtidig gi en algoritme som finner maksimum strøm. I tillegg finner den et minimum kutt (altså et kutt med minimum kapasitet). Dette høres lovende ut, så la oss gå igang med arbeidet!

Eksempel: Se på eksemplet i Figur 3.3 der alle kapasiteter er 6. Den angitte strømmen har verdi 12. Dette er ikke maksimum strøm fordi vi kan øke strømmen i (s, u) til 6 og strømmen i (u, t) til 5. Da får vi en ny st-strøm (hvorfor?) og verdien er 11. Er dette maksimum strøm? □

Legg merke til at det finnes en enkel type st-strøm. La P være en rettet st-vei og la $\epsilon > 0$. Definer en st-strøm x ved at $x_{u,v} = \epsilon$ for hver rettet kant $(u, v) \in P$ og $x_{u,v} = 0$ ellers. Det er klart at dette virkelig er en st-strøm og verdien er $v(x) = \epsilon$.

Mer generelt kan vi se på en vei P fra s til en node v, der veien ikke nødvendigvis er rettet. Dermed vil de rettede kantene i P falle i to grupper: *foroverkanter* som har retning fra r mot v, og *bakoverkanter* som har retning fra v mot r. Anta x er en st-strøm og at rv-veien P oppfyller at

(i) $x_{u,v} < c_{u,v}$ for hver foroverkant i P, og
(ii) $x_{u,v} > 0$ for hver bakoverkant i P.

Da kaller vi P en *x-tilføyende vei til v*. Poenget med dette begrepet er følgende:

- *hvis P er en x-tilføyende vei til t, og vi øker strømmen med en passende liten $\epsilon > 0$ på alle foroverkanter i P og reduserer med ϵ på alle bakoverlinjer i P, så får vi en ny st-strøm med verdi $v(x) + \epsilon$.*

Så ved å finne en x-tilføyende vei kan vi finne en ny strøm med høyere verdi. Deretter kan vi gjenta prosessen og øke strømmen ytterligere. Og for å komme igang bruker vi den trivielle strømmen $x = 0$. Dette er idéen! Og her er et viktig resultat som sier oss at dette er en metode som virker.

Lemma 3.6. *En st-strøm x en en maksimum strøm hvis og bare hvis det ikke finnes noen x-tilføyende vei til t.*

Bevis: Let x være en st-strøm og la $S(x)$ være mengden av de noder v slik at det finnes en x-tilføyende vei til v. Da er det to mulighter.

Tilfelle 1: $t \in S(x)$. Da fins det en x-tilføyende vei til t og vi kan finne en ny st-strøm med høyere verdi (se over). Da er ikke x en maksimum strøm.

Tilfelle 2: $t \notin S(x)$. Da vil enhver kant (u, v) der $u \in S(x)$ og $v \notin S(x)$ oppfylle at $x_{u,v} = c_{u,v}$. For hvis $x_{u,v} < c_{u,v}$, ville det finnes en x-tilføyende vei til v (via u). Videre vil enhver kant (u, v) der $u \notin S(x)$ og $v \in S(x)$ oppfylle at $x_{u,v} = 0$. For hvis $x_{u,v} > 0$, ville det finnes en x-tilføyende vei til v (via u). Dermed får vi (ved strømbevaring)

$$\begin{aligned} v(x) = \; & \textstyle\sum_{v \in S(x)} (x(\delta^+(v)) - x(\delta^-(v))) = x(\delta^+(S(x))) - x(\delta^-(S(x))) = \\ & x(\delta^+(S(x))) = c(\delta^+(S(x))). \end{aligned}$$

Men da har kuttet $\delta^+(S(x))$ kapasitet lik verdien av strømmen x, så ved Lemma 3.5 må da x være en maksimum st-strøm og $\delta^+(S(x))$ være et minimum st-kutt. □

Vi er nå framme ved *maks strøm - min kutt teoremet* som ble oppdaget i 1956, uavhengig av hverandre, av Ford og Fulkerson og av Elias, Feinstein and Shannon.

Teorem 3.1. *For enhver rettet graf med kapasitetsfunksjon c på kantene og med distinkte noder s og t vil verdien av maksimum strøm være lik minimum kutt kapasitet.*

Bevis: Vi vet fra Lemma 3.3 at en maksimum st-strøm virkelig finnes. Let x be a maximum flow which is known to exist by Lemma 3.3. Ved Lemma 3.6 er det da ingen x-tilføyende vei til t og verdien $v(x)$ er lik kapasiteten til kuttet $\delta^+(S(x))$. Dermed følger det også av Lemma 3.5 at $\delta^+(S(x))$ er et minimum st-kutt. □

Dette resultatet er et eksempel på et *minmaks-teorem*: maksimumsverdien av en viss funksjon over en viss mengde er lik mimimumsverdien av en annen funksjon tatt over en annen mengde. Det er en lang rekke resultater i kombinatorisk optimering som er av denne typen.

En generisk algoritme for å løse maksimum strøm problemet og dessuten minimum kutt problemet blir da som følger (med notasjon $S(x)$ som over).

Generisk maks-strøm algoritme.

0. La $x = 0$.
1. Bestem node mengden $S(x)$. Hvis $t \notin S(x)$, stopp. Da er x en maksimum st-strøm og $\delta^+(S(x))$ er et minimum st-kutt. Ellers, la P være en x-tilføyende vei til t.
2. Oppdater strømmen x ved å øke strømmen med ϵ på hver foroverkant i P og redusere med ϵ på hver bakoverkant i P. Her velges ϵ størst mulig uten at kapasitetsbegrensningene brytes. Returner til steg 1.

Denne algoritmen vil terminere når kapasitetene er rasjonale tall. En viktig konsekvens av algoritmen er at det finnes en *heltallig* maksimum strøm når alle kapasitetene er heltallige!

Korollar 3.1. *Når alle kapasitetene er heltallige vil den generiske maksimum strøm algoritmen terminere med en heltallig maksimum strøm, dvs. $x_{u,v}$ er et heltall for hver $(u, v) \in E$.*

Bevis: I hver iterasjon vil vi få at ϵ blir et heltall, og siden vi starter med $x = 0$ vil alle strømmer vi finner være heltallige. □

En viktig konsekvens av maks strøm - min kutt teoremet er *Menger's theorem* (fra 1927, see [12]). Dette er et sentralt resultat i kombinatorikk som involverer konnektivitet i grafer. Vi sier at en mengde med rettede st-veier i en rettet graf er *kant-disjunkte* hvis hver kant i grafen tilhører høyst én av disse veiene.

Teorem 3.2. *Maksimum antall kant-disjunkte st-veier i en rettet graf er lik det laveste antall kanter i et st-kutt.*

Vi lar det være en oppgave å vise dette resultatet ut fra maks strøm - min kutt teoremet.

Forslag til videre lesning. En glimrende innføringsbok i kombinatorikk der man finner en del grafteori er Brualdi's bok [5]. En bok som dekker mange av de samme temaene og diskret matematikk mer generelt er [11]. Ellers finnes en mengde bøkre i grafteori, f.eks. [8], [2],[6] og [10].

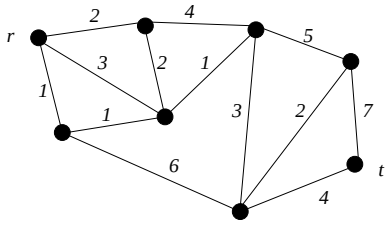
En anbefalt bok i om grafoptimering og strøm i nettverk er [1]. Der finner man både detaljer om algoritmer, teori og anvendelser. En annen bok som dekker mange av de samme temaene er [3].

En hefte om kombinatorisk optimering og konveksitet som har vært i bruk i et hovedfagskurs i optimering ved UiO er [7].

En meget god innføringstekst i lineær programmering er boken [13] (som også er i bruk i et annet kurs i optimering ved UiO).

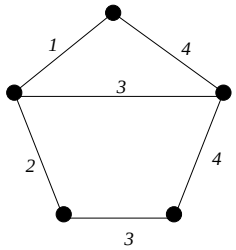
3.4 Oppgaver

1. Løs korteste vei problemet i Figur 3.4.



Figur 3.4: En korteste vei oppgave.

2. Betrakt grafen i Figur 3.5. Bestem den tilhørende metrikken.



Figur 3.5: En metrikk oppgave.

3. Betrakt korteste vei problemet i en graf $G = (V, E)$, der l er en gitt ikke-negativ lengdefunksjon og avstandene beregnes fra noden r. Anta at vi kjenner avstandene fra r til en viss mengde av noder T. Hvordan kan du bruke denne informasjonen til å finne avstanden fra r og til enhver node i $V \setminus T$?

4. Hvordan ville du løse *alle-par korteste vei problemet* der man skal finne avstanden mellom hvert par av noder i en vektet graf?

5. La $G = (V, E)$ være en vektet graf med $l_e \geq 0$ for alle $e \in E$. Anta at vi har et tall d_v for hver node $v \in V$ (dvs. en funsjon $d : V \to \Re$) som oppfyller $d_v \leq d_u + l_{uv}$ for alle $[u, v] \in E$. La videre P være en rt-vei som oppfyller at $d_v = d_u + l_{uv}$ for alle $[u, v] \in P$. Vis at da er P en korteste rt-vei.

6. Vi skal se på et viktig problem i pålitelighetsanalyse (et område i statistikk og sannsynlighetsregning). Betrakt en graf $G = (V, E)$ og la s og t være to noder i grafen. La p_e være et gitt tall for hver kant e der vi har at $0 < p_e \leq 1$. Vi kan tenke på p_e som pålitelighten til e, dvs. sannsynligheten for at kanten er operativ ("virker"). Påliteligheten til en st-vei er definert som produktet av påliteligheten til hver av kantene i veien. Dette er (under antagelse om uavhengighet) det samme som sannsynligheten for at veien P er operativ. Hvordan kan man finne en mest pålitelig st-vei? (Hint: se på logaritmen til p_e.)

7. I en rettet graf $D = (V, E)$ vis at

$$\sum_{v \in V} |\delta^+(v)| = \sum_{v \in V} |\delta^-(v)|.$$

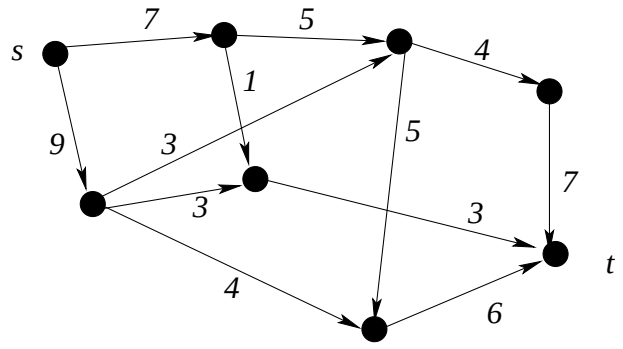
8. La $D = (V, E)$ være en rettet vei og la s og t være to noder i D. Forsøk å gi en algoritme som finner en rettet st-vei eller slår fast at en slik vei ikke finnes.

9. Betrakt en st-strøm x i et nettverk. Vis at strømmen ut fra node s er lik strømmen inn til node t.

10. Betrakt den rettede graf med kapasiteter knyttet som er vist i Figur 3.6. Forsøk å finne en maksimum st-strøm og et minimum st-kutt ved inspeksjon.

11. Løs samme problemer som i forrige oppgave, men bruk istedet den generiske maks strøm algoritmen.

12. Betrakt følgende problem innen distribuerte beregninger. Vi har en datamaskin med to prosessorer. Det er gitt en mengde J av jobber og vi skal for hver jobb $j \in J$ avgjøre om denne bør sendes til prosessor 1 eller prosessor 2 (jobben skal bare kjøres på én av maskinene). Følgende data er gitt. Regnetiden for jobb j på prosessor 1 (2) er α_j (β_j). Videre er det kommunikasjonsbehov som oppstår mellom de to prosessorene: hvis jobb i og



Figur 3.6: En maksimum strøm oppgave.

jobb j går på forskjellige maskiner er kommuniksjonen $c_{i,j}$. Kvaliteten på en jobballokering defineres som summen av regnetidene for jobbene (på den maskinen hver enkelt allokeres til) og det totale kommunikasjonsbehovet. Dette problemet kan modelleres på følgende vis. Lag en rettet graf med nodemengde $V = \{s, t\} \cup J$ og rettede kanter: (s, j) og (j, t) for hver $j \in J$ samt (i, j) for hver $i, j \in J$ der $i \neq j$. Bestem kapasiteter på alle kantene slik at allokeringsproblemet svarer til minimum st-kutt problemet i denne grafen. (Hint: definer kapasiteten $c_{i,j}$ på hver rettet kant (i, j) der $i, j \in J$, $i \neq j$).

13. Betrakt maksimum strøm problemet i en rettet graf der alle kapasiteter er 1. Lag et "passende stort" eksempel og bruk den generiske maksimum strøm algoritmen. Hvordan ser den optimale løsningen ut? Hva med minimum kutt?

14. Vis Menger's teorem (Teorem 3.2) ut fra maks strøm - min kutt teoremet.

Bibliografi

[1] R. K. Ahuja, T. L. Magnanti, and J. B. Orlin. *Network flows: theory, algorithms, and applications*. Prentice-Hall, Englewood Cliffs, New Jersey, 1993.

[2] C. Berge. *Graphs and Hypergraphs*. North-Holland, Amsterdam, 1973.

[3] D.P. Bertsekas. *Linear network optimization: algorithms and codes*. MIT-Press, 1991.

[4] N.L. Biggs, E.K. Lloyd, and R.J. Wilson. *Graph theory 1736–1936*. Oxford: Clarendon Press, 1998.

[5] R.A. Brualdi. *Introductory combinatorics. 3rd ed.* Prentice-Hall, 1999.

[6] G. Chartrand and L. Lesniak. *Graphs and digraphs*. Wadsworth and Brooks, 1986.

[7] G. Dahl. An introduction to convexity, polyhedral theory and combinatorial optimization. Technical Report 67, University of Oslo, Institute of Informatics, Oslo, Norway, Jan.1997.

[8] R. Diestel. *Graph theory*. Springer, 2000.

[9] L. Euler. Solutio problematis ad geometriam situs pertinensis. *Commentarii Academiae Petropolitanae*, 8:128–140, 1736.

[10] M. Gondran and M. Minoux. *Graphs and algorithms*. Wiley, 1984.

[11] R. Grimaldi. *Discrete and combinatorial mathematics*. Addison Wesley, 1994.

[12] K. Menger. Zur allgemeinen kurventheorie. *Fundamenta Mathematicae*, 10:96–115, 1927.

[13] R.J. Vanderbei. *Linear programming: foundations and extensions*. Kluwer, 1996.

Register