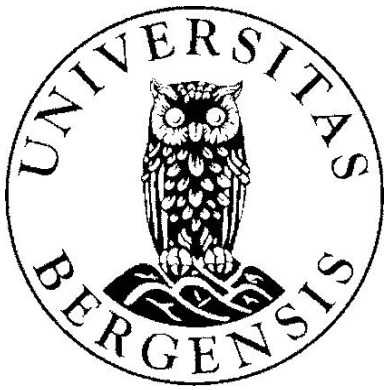
Naturlige nettverk

Martin Vatshelle

7. mars 2008



Masteroppgave i Algoritmer
ved
Universitetet i Bergen

Sammendrag

Hoveddelen av denne Masteroppgaven er en innføring i naturlige nettverk. Vi forklarer og diskuterer begrepet naturlige nettverk og vurderer hvilke nettverk som bør kalles naturlige. Vi ser på forskjeller og likheter mellom tilfeldige, planlagte og naturlige nettverk.

Vi gir en innføring i "verden er liten" fenomenet og beskriver en modell gitt av Kleinberg [14] som kan forklare fenomenet.

Vi ser på strukturell balanse og vurderer kompleksiteten for å avgjøre om et nettverk har strukturell balanse. Vi prøver å finne et mål på hvor ubalansert et nettverk er og vurderer kompleksiteten til flere alternative mål for strukturell balanse.

Vi avslutter med hva som kan gjøres for å forbedre kjøretiden på algoritmer der naturlige nettverk er del av innput. En parameter for rettede grafer som kan bli viktig i denne sammenheng er Kelly-bredde. Siste del av oppgaven er opptrykk av en artikkel, på engelsk, som gir den første polynomiske algoritmen for gjenkjenning av rettede grafer med Kelly-bredde 2.

Forord

Arbeidet med min masteroppgave startet med et prosjekt om Kelly-width sammen med Daniel Meister og min veileder Jan Arne Telle. Jeg var med fra begynnelsen av prosjektet og fulgte hele utviklingen helt til vi fikk en artikkel godkjent ved "33^{rd} International Workshop on Graph-Theoretic Concepts in Computer Science" (WG). Jeg reiste til Jena i juni 2007 og presenterte arbeidet vårt på WG. Artikkelen har senere blitt omskrevet til en mye kortere og enklere versjon enn den vi fikk godkjent til WG. Den endelige versjonen er vedlagt oppgaven min som et appendiks og har blitt sendt til tidsskriftet Discrete Applied Mathematics. Jeg kan trygt si at jeg har hatt stor innflytelse på den ferdige artikkelen og at jeg har fulgt hele prosessen med å publiserere en vitenskapelig artikkel.

Hoveddelen av oppgaven er en introduksjon til naturlige nettverk. Jeg håper at denne teksten kan være begynnelsen på et kompendium som eventuellt kan brukes i kurs-sammenheng og vekke en viss interesse for dette spennende fagfeltet. David Easley og Jon Kleinberg underviser nå et slikt kurs ved Cornell University i USA [6]. Det kursmateriell som har blitt gitt ut i forbindelse med dette kurset har vært til stor hjelp for meg, og blitt brukt som modell for arbeidet. Mange resultater som blir presentert her finnes også i kursmaterialet som blir brukt av David Easley og Jon Kleinberg. Kompleksitetsresultatetene for de fire balanseindeksene i kapittel 3.2 har jeg utviklet på egenhånd. Jeg har ikke klart å finne referanser i litteraturen til tilsvarende resultater.

Jeg ble interessert i strukturell balanse da jeg kom i kontakt med førsteamanuensis Tom Backer Johnsen ved Institutt for samfunnspsykologi ved UiB. Han samlet data om forhold innen flere grupper av soldater og fulgte utviklingen over tid. Meningen var å finne ut om vennskapsforholdene i en gruppe utviklet seg slik at gruppen kom nærmere strukturell balanse etter hvert som personer ble bedre kjent med hverandre. Han ønsket en algoritme som måler den strukturelle balansen og å finne ut hvor vanlig/uvanlig et gitt mål er. Dette er bare et av de mange temaene innen naturlige nettverk som er interessant både for sosiologi og informatikk og åpner for tverrfaglig samarbeid.

Innhold

1 Introduksjon **1**
1.1 Oversikt over kapitlene 5

2 Hva er et naturlig nettverk? **6**
2.1 Fysiske begrensninger 6
2.2 Tilfeldige nettverk 7
2.3 Planlagte nettverk 10
2.4 Webgrafen 11
2.4.1 Samle informasjon om webgrafen 11
2.4.2 Webgrafen er ikke tilfeldig 11
2.4.3 Webgrafen er ikke planlagt 14

3 Strukturell balanse **16**
3.1 Definisjon av strukturell balanse 16
3.2 Mål for strukturell balanse 21

4 Verden er liten **28**
4.1 Introduksjon 28
4.2 Korte stier 29
4.3 Finne korte kjeder i et nettverk 29

5 FPT og trelignende grafer **34**
5.1 Trebredde 35
5.2 Kelly-bredde 36

6 Oppsummering og videre arbeid **38**

Appendiks **41**

A Definisjoner **41**

B Recognizing digraphs of Kelly-width 2 **43**

1 Introduksjon

Et nettverk er en samling objekter der noen par er forbundet med hverandre. For eksempel er ethvert menneske en del av et nettverk der man er forbundet med alle sine venner, dette kalles et vennskapsnettverk. Objektene i et nettverk kan være hva som helst, mennesker, internettsider eller radiomaster. Et nettverk av stor interesse for reklamefirma og søkemotorer er webgrafen, der er objektene internettsider og forbindelsene hyperlenker.

Studiet av naturlige nettverk fra et matematisk og algoritmisk ståsted er et nytt fenomen og de grunnleggende terminologiske bestanddelene er fortsatt gjenstand for diskusjon. Selv ikke for spørsmålet om hva som kjennetegner et naturlig nettverk har man oppnådd konsensus omkring et entydig svar. Vi ønsker å forstå hvilke faktorer som bestemmer hvilke par som er forbundet i et naturlig nettverk. Når vi forstår dette kan vi lage modeller som kan hjelpe oss å finne svar på mange viktige spørsmål.

Kan vi dele inn i grupper etter type objekter? Det er svært vanlig å dele inn etter type objekt, er objektene personer bedrifter eller organisasjoner har du et sosialt nettverk. Er det snakk om datamaskiner så kalles det datanettverk, LAN, WAN eller lignende. Så har vi webgrafen og en hel rekke andre. Disse igjen kan deles inn i undergrupper. Denne typen kategorisering gir mange grupper og mange egenskaper finnes på tvers av gruppene. Det er derfor vanskelig å lage en komplett klassifisering på denne måten, men det er en god måte å avgrense feltet på og nødvendig når data skal samles, men den hjelper oss ikke å forstå nettverkene, bare beskrive dem. Vi ønsker å finne egenskaper som skiller naturlige nettverk fra andre nettverk. Det finnes par av nettverk som har samme type objekter men der et er naturlig og det andre ikke er naturlig.

Kan vi dele inn i grupper etter type relasjoner? Relasjoner kan være mye, for eksempel er alle vennskap forskjellig, ikke bare fordi de består av forskjellige personer, de endres også med tiden. Bare det å prøve å måle og beskrive et vennskap er vanskelig og utenfor denne oppgaven. Vennskap for eksempel måles på en skala fra 1-10, dette kan også inngå i et nettverk. Det er derfor vanskelig å holde styr på alle typer av relasjoner. Det kan også være typer av relasjoner som finnes i både naturlige nettverk og andre nettverk.

For å belyse spørsmålet om hva som kjennetegner et naturlig nettverk velger vi å innføre en tredeling av nettverkene. Vi kan si at det er tre drivkrefter bak et nettverk, planlegging, tilfeldighet og utvikling.

- Planlegging er når hele nettverket tas i betraktning og de koblingene som er best for helheten finnes, for eksempel et datanettverk eller et veinett. At det er en sentral styring som har oversikt over hele nettverket fører til en spesiell struktur som sjeldent oppstår ellers og regnes ikke for naturlig.

- Tilfeldighet er en viktig del i de fleste nettverk, men nettverk som kun styres av tilfeldighet finnes sjeldent i virkeligheten. Slike tilfeldige nettverk brukes ofte i teoretiske analyser og simuleringer. Tilfeldige nettverk er ikke naturlige.

- Utvikling er når forbindelser oppstår på grunn av lokale valg, det vil si at hvert objekt (par av objekter) har mulighet til å danne en forbindelse uten en sentral styring. Hvert objekt har kun begrenset informasjon om nettverket og vet lite eller ingenting om objekter som det selv ikke er forbundet med.

Naturlige nettverk ser ut til å styres av utvikling, men kan være sterkt påvirket av tilfeldighet. Hvis planlegging er en viktig del av nettverket er det ikke naturlig.

Datanettverket på kontoret er ikke et naturlig nettverk. Det er planlagt og konstruert på forhånd for å gi best funksjonalitet. De enkelte brukerne kan ikke endre nettverket, og nesten ingenting er overlatt til tilfeldighetene. Bryter en av koblingene sammen vil den straks repareres. Også for et veinett gjelder det at veier blir reparert når de blir ødelagt. Selv om ikke alle koblingene er planlagt på forhånd, sees det på nettverket som en helhet og det veies opp fordeler og ulemper før et kompromiss inngåes og en ny kobling opprettes.

I et vennskapsnettverk oppstår nye vennskap hele tiden og gamle brytes. Det er ingen sentral styring som påvirker opprettelsen av et vennskap, det er personene selv som oppretter vennskapene. Derfor spiller planlegging liten rolle mens utvikling er viktig. Det er mye som er tilfeldig, hvem du møter på ferietur eller hvilken klasse du havner i. Men det er hvordan personer oppfatter og påvirker hverandre som er styrende for nettverkets struktur.

Webgrafen er delvis planlagt men fortsatt naturlig. Akkurat som i et vennskapsnettverk er det eieren av hver enkelt side som bestemmer hvilke andre sider den skal ha lenker til. Men når du bestemmer deg for å lage en lenke fra din side til en annen har du informasjon om mye mer enn de sidene du allerede har lenket til. Dette innfører et element som gjør webgrafen forskjellig fra et vennskapsnettverk. En del planlegging inngår også i utviklingen av

webgrafen. Store firma har gjerne mange servere og internettsider der hyperlenkene er nøye planlagt (lokal planlegging). Det viktigste er likevel tema og språket på websidene. Du finner gjerne sidene du velgen a lenke til ved å følge tilfeldige lenker til du ender opp på en side du liker. Selv om en del er planlagt er mye tilfeldig og mye styrt av utvikling, derfor regnes webgrafen som et naturlig nettverk.

Kan vi dele nettverk inn i grupper etter hvordan relasjoner oppstår og forsvinner? Når vi ser på vennskap så er det lett å observere at to personer som bor nær hverandre geografisk oftere blir venner enn to som bor langt fra hverandre. Når den geografiske avstanden blir stor er det ikke lenger en avgjørende faktor, da er personlige egenskaper som yrke, kjønn og alder mye viktigere. Hvis en webside har en lenke til en annen side er det ikke avgjørende hvor stor den geografiske avstanden er, men om de er på samme språk og om samme tema. I noen nettverk oppstår eller brytes relasjoner ofte (ustabile nettverk) mens andre er langsommere (stabile nettverk). Egenskaper som geografisk avstand er veldig viktig for naturlige nettverk, hvis vi klarer å finne ut hvilke slike egenskaper som er avgjørende vil det bli lettere å se hva som skiller de naturlige nettverkene fra andre nettverk. Men slike egenskaper i seg selv helper lite for å avgjøre om et nettverk er naturlig eller ikke, de gjelder for andre typer av nettverk også.

I et vennskapsnettverk vil ofte dine venner presentere deg for sine venner, slik at sannsynligheten for nye vennskap ikke bare er avhengig av den informasjonen du har men også dine venners informasjon. Her ser vi et eksempel på at nettverkets struktur påvirker hvordan koblingene oppstår. Det er ikke lenger snakk om rene sannsynligheter som kun påvirkes av objektenes egenskaper. Sannsynligheten for relasjoner er sterkt avhengig av hvilke relasjoner som allerede finnes og endrer seg hele tiden. Dette er de egenskapene vi håper å kjenne igjen og skille fra hverandre og tror det er slike egenskaper som skiller naturlige nettverk fra andre nettverk. Dette eksempelet involverer kun 3 personer og kan ganske lett oppdages, det er vanskeligere å oppdage slike sammenhenger med mange objekter, men de kan godt finnes.

Søkemotorer er avhengig av å gi relevante resultater for å bli populære, og webgrafen er den informasjonen de har. Blir en søkemotor populær får den mer reklameinntekter. Reklamefirmaer ønsker å vite hvor populær søkemotoren blir i fremtiden slik at de kan være først ute med tilbud og dermed spare penger. Dette avhenger av hvor godt brukerne liker søkemotorens resultater og er dermed avhengig av hvordan webgrafen utvikler seg. Hvis vi kan gjenkjenne strukturer og drivkrefter bak webgrafen så kan det hjelpe

oss til å forstå webgrafen. Dette vil blant annet åpne muligheter for bedre søkemotorer. Finner vi nyttig struktur i webgrafen kan det være at samme metode vil være nyttig for andre naturlige nettverk.

1.1 Oversikt over kapitlene

- Kapittel 2 tar for seg spørsmålet om hva et naturlig nettverk er. Siden det ikke er enighet om en definisjon av naturlige nettverk ser vi på noen eksempel som vi vet ikke er naturlige og vi ser noen eksempler som bør falle innenfor en definisjon av naturlige nettverk. Spesielt ser vi på webgrafen.

- Kapittel 3 handler om strukturell balanse. Først definerer vi strukturell balanse og finner en algoritme for å avgjøre om et nettverk har strukturell balanse. Deretter vurderer vi 4 metoder for å bestemme hvor nær balanse et nettverk er. For hver av de 4 metodene diskuterer vi kompleksiteten og sammenligner de med hverandre for å prøve å vurdere hvilken metode som er best.

- Kapittel 4 gir en innføring i "verden er liten" fenomenet og beskriver en modell som er lagt frem av Jon Kleinberg.

- Kapittel 5 Gir en liten innføring i parametrisert kompleksitet og viser hvordan dette kanskje kan hjelpe oss å finne bedre algoritmer for problemer hvor svaret avhenger av et naturlig nettverk. Spesielt er Kelly-bredde nevnt som en slik parameter.

- Kapittel 6 er en liten oppsummering og forslag til videre arbeid.

- Appendiks A inneholder definisjoner av en del uttrykk som brukes i teksten, vi valgte å plassere dette som et appendiks siden det får teksten til flyte bedre.

- Appendiks B er et opptrykk av en artikkel om Kelly-bredde.

2 Hva er et naturlig nettverk?

Vi har ikke enighet om en definisjon av naturlige nettverk, men mange nettverk er naturlige og mange er ikke. Vi skal se på noen nettverk som er naturlige og noen som ikke er naturlige, men det vil ta tid før vi har en enighet om nøyaktig hva et naturlig nettverk er. Derfor blir fokus i dette kapittelet mer på hva vi vet og hva vi ønsker å finne ut.

Det er mange problemer der et naturlig nettverk er del av inndata. For eksempel ønsker reklamefirmaer å spre reklame slik at den når flest mulig personer som er i den rette målgruppen. Hvis en reklame når frem til en person vil personen kanskje spre budskapet videre til sine venner. Derfor er det kanskje lurt å spre reklamen over et stort område og håpe at personene som mottok reklamen sprer den videre, men reklame virker sterkere hvis den sees mange ganger og hvis flere venner også har hørt om det produktet som reklamen sikter til. Derfor kan det være lurt å sikte mot en spesiell målgruppe i et mindre område. Det er en komplisert oppgave å finne den beste måten å spre reklame på.

For å teste ut forskjellige strategier for spredning av reklame kan det brukes et nettverk der nodene er personer og hvert ordnet par av personer (a, b) har en vekt som sier hvor mye a påvirker b. Dette regnes som et naturlig nettverk. Det er mange andre ting enn reklame som spres i nettverk. Rykter, datavirus, trafikkork, sykdommer og adferdsmønstre som mobbing er noen eksempler. Kostnaden ved å samle inn data om et nettverk overstiger fort gevinsten en god modell kan gi, derfor trengs en enkel modell som samtidig har egenskaper som ligner de nettverkene vi ønsker å studere.

Målet er å finne en modell som kan brukes til å simulere naturlige nettverk og samtidig gjøre problemer som for eksempel spredning av reklame lettere å løse. I dette kapittelet skal vi diskutere hva som kjennetegner et naturlig nettverk, og hvilke andre typer nettverk som finnes. Det er ikke noe entydig svar på hva som er et naturlig nettverk, derfor konsentrerer vi oss om å finne noen nettverk som er naturlige og noen som ikke er naturlige. Først når vi forstår strukturen og egenskapene til naturlige nettverk kan vi begynne å utvikle gode modeller. Et av de viktige nettverkene vi skal se på er webgrafen, et nettverk vi ønsker å ha en god modell for.

2.1 Fysiske begrensninger

Fysiske begrensninger er en av mange faktorer som påvirker hvordan et nettverk ser ut. De fleste nettverk består av objekter som befinner seg på

jordens overfate og den fysiske avstanden mellom objektene påvirker ofte sannsynligheten for at de er forbundet. For eksempel er det mer sannsynlig at det går en direkte vei mellom to byer som er nær hverandre, enn mellom to byer som er langt fra hverandre. Samme gjelder for bekjentskap, det er mer sannsynlig å kjenne en person hvis han er en nabo enn om han bor langt vekke. Det kan også være hindringer mellom objektene, mobiltelefoner og radioer er avhengig av sendere som ikke har for mange hindringer mellom sender og mottaker. Ikke alle nettverk er avhengig av den geografiske avstanden, særlig blir det tydelig over Internett. Mange personer møtes via Internett fordi de har felles interesser, det kan være et spill eller en faglig diskusjon. Nettverket av vennskap som oppstår via Internett er dermed ikke avhengig av avstand men av felles interesser (litt avhengig av tidssoner).

Hvert objekt kan ha begrensninger på hvor mange forbindelser det kan ha, mennesker har begrensning på hvor mange venner det går an å holde kontakten med, datamaskiner og rutere har begrensning på hvor mange linjer de kan kommunisere med. Det samme gjelder forbindelser, disse kan som regel ikke finnes mellom alle typer av objekter, for eksempel er det vanskelig for to personer å bli venner hvis de ikke snakker et felles språk. Egenskapene til objektene er med på å avgjøre sannsynligheten for en kobling mellom dem.

Felles for alle disse begrensningene er at de kan påvirke et nettverk uansett om det er tilfeldig, planlagt eller naturlig. Strukturer som skyldes slike fysiske egenskaper er ganske lette å bekrefte hvis den riktige egenskapen undersøkes, problemet er at det er så mange ting som kan påvirke nettverket og derfor vanskelig å finne den rette egenskapen å undersøke. Mange egenskaper er avhengig av typen på nettverket og det er så mange forskjellige typer nettverk at det kan bli vanskelig å finne generelle sammenhenger som skiller naturlige nettverk fra andre nettverk.

2.2 Tilfeldige nettverk

Med tilfeldige nettverk mense nettverk som kan beskrives av en tilfeldig graf. En tilfeldig graf er en graf der noder og kanter er generert tilfeldig. Hvert par har en gitt sannsynlighet for at den finnes en kant mellom dem. Sannsynligheten er vanligvis uavhengig av hvilke andre par det finnes kanter mellom. Denne grafmodellen er veldig nyttig i mange tilfeller fordi det er lett å generere slike grafer. For å generere tilfeldige grafer, genereres først nodene, så brukes en funksjon som for hvert par av noder gir en sannsynlighet, og med gitt sannsynlighet opprettes en kant mellom nodene.

Det finnes også en annen modell for tilfeldige grafer. Gitt noen egenskaper, velges det tilfeldig blant alle grafer som har disse egenskapene. Det kan være litt uklart hvilke egenskaper vi har lov til å velge, om det er begrensninger på hvor mange og hvilke typer. Noen eksempler er: et tilfeldig tre, en tilfeldig sammenhengende graf eller en tilfeldig graf der alle nodene har gradtall 7. Legger vi for mange begrensninger på grafen blir det for lite tilfeldighet og grafen ligner mer på en planlagt graf.

Den vanligste, enkleste og mest studerte modell av tilfeldige grafer er Erdős-Rényi modellen, oftest kalt $G(n, p)$ der n er et positivt heltall og $0 < p < 1$. Letteste måte å definere denne modellen på, er å gi en algoritme for å generere grafer av type $G(n, p)$.

1. Lag en graf med n noder og ingen kanter.
2. For hvert par (a, b) av noder, med sannsynlighet p, legg til en kant (a, b).

For å regne på sannsynligheter er det viktig å innse forskjellen på merkede grafer og grafer. I merkede grafer har alle nodene fått et unikt nummer mellom 2 og n. I en graf der alle nodene er like vil sykler kun kunne skilles fra hverandre ved forskjellig lengde, mens i en merket graf må to sykler i tillegg til å ha like mange noder også ha den samme merkingen av nodene og nodene må komme i samme rekkefølge i syklene.

I modellen $G(n, 0.5)$ har ethvert par av noder samme sannsynlighet for at det er en kant mellom dem som at det ikke er en kant mellom dem. Det vil si at alle de $2^{\binom{n}{2}}$ merkede grafene på n noder har lik sannsynlighet for å bli generert. Det kan være lett å tenke seg at en graf av type $G(n, 0.5)$ dermed ikke har noe struktur. Men noen egenskaper finnes i mange merkede grafene mens andre finnes i få merkede grafer. I tillegg er det forskjell på hvor mange forskjellige merkede grafer som blir samme graf hvis vi fjerner merkingen. Dette gjør at mange egenskaper kan forventes i en graf generert av $G(n, 0.5)$.

Når vi ser på $G(n, 0.5)$ har alle merkede grafer lik sannsynlighet for å bli generert. For mange egenskaper er det mulig å regne ut sannsynligheten for at en graf generert av $G(n, 0.5)$ har denne egenskapen. Hvis det er en egenskap som, for store n, finnes med sannsynlighet tilnærmet lik 1 sier vi at nesten alle grafer har denne egenskapen. Formelt må en egenskap finnes i en graf generert av $G(n, 0.5)$ med sannsynlighet større enn $1 - \varepsilon$, for enhver konstant $0 < \varepsilon$. Mange egenskaper er kjent å gjelde for nesten alle grafer, som for eksempel: nesten alle grafer er sammenhengende.

Hvilke grafer som mest sannsynlig blir generert av $G(n,p)$ avhenger sterkt av p. Hvis $p = 2/n$ vil forventet antall kanter være $\binom{n}{2} * p = n-1$. Den eneste måten en graf på n noder med $n-1$ kanter kan være sammenhengende er hvis grafen er et tre. Det er derfor lite sannsynlig at $G(n, 2/n)$ er sammenhengende. Hvis sannsynligheten økes til $p > \log_2(n)/n$, blir det veldig sannsynlig at grafen er sammenhengende. For store verdier av n blir $\log_2(n)/n$ mindre enn enhver konstant, derfor sier vi at for enhver konstant $0 < p < 1$ er nesten alle grafer av typen $G(n,p)$ sammenhengende.

En graf generert av denne modellen kan gjenkjennes med stor sannsynlighet. Forventet gradtall i en graf av typen $G(n,p)$ vil være $p * (n-1)$. Sannsynligheten for at gjennomsnittlig gradtall avviker mye fra dette blir veldig liten når n blir stor. Dette gjør det enkelt å tilnærme p for en graf som er laget av en slik tilfeldig modell. Siden alle noder har lik sannsynlighet for å være nabo til en gitt node v vil gradtallene følge en binomisk fordeling. Siden gradtallene er binomisk fordelt og vi kan tilnærme p kan vi sjekke om gradtallene i en graf tilnærmet passer denne fordeling. Hvis de ikke gjør det kan vi med stor sikkerhet fastslå at den ikke er generert av en slik tilfeldig modell. En annen egenskap er at siden alle kanter har lik sannsynlighet, så vil en indusert delgraf på k noder være en graf av type $G(k,p)$. Vi har derfor 2^n delgrafer som alle er tilfeldige, dette gjør at vi får et godt grunnlag for å vurdere om en graf kan være generert av Erdős-Rényi modellen.

Selv om de fleste naturlige nettverk ikke kan beskrives med Erdős-Rényi modellen betyr ikke det at de ikke er tilfeldige. Dette fordi sannsynligheten for at en kant finnes ikke trenger å være lik for alle kanter. Det er krevende å vise at ingen tilfeldig modell kan beskrive et gitt nettverk. Hvis sannsynligheten for at en kant finnes aldri er 0 eller 1, kan enhver tilfeldig modell med en liten sannsynlighet faktisk gi et hvert nettverket. Vi kan kun beregne sannsynligheten for at et gitt nettverk er tilfeldig. Hvis disse nettverkene for eksempel er vennskap mellom elever i en skoleklasse, så er det klart at modellen avhenger av antall elever, men det er også mange andre faktorer som kan spille inn, som for eksempel:

- Antall jenter i forhold til gutter.
- Alder på elevene.
- Om skolen er i en by eller på landet.
- Hvor mange klasser det er på skolen.

Før vi vet alle disse faktorene kan vi ikke få en nøyaktig modell, og det er antagelig ikke mulig å få oversikt over alle faktorene. Derfor prøver vi å finne de faktorene som er viktigst og så finne en funksjon som basert på disse faktorene finner en sannsynlighet for en kant mellom et gitt par. Et tilfeldig nettverk skal være lett å generere, av regler som kan generere noder og kanter uten å være for opptatt av strukturen til den delen som allerede er generert. Det vil si at enten må egenskaper være bestemt av vennskapene til en person eller så må vennskapene være bestemt av egenskapene. Men i virkeligheten skjer begge deler, det er derfor vanskelig å beskrive naturlige nettverk med en tilfeldig modell.

Tilfeldige grafer brukes ofte som modeller i virkeligheten fordi de er lett å generere, og andre modeller som er bedre vil være tidkrevende og kostbare å utvikle. Hvis det går an å finne en generell modell som er enkel å generere og ligner mer på naturlige nettverk enn en tilfeldig graf ville det vært til stor nytte.

2.3 Planlagte nettverk

Nå som vi har begrunnet at ikke alle nettverk er tilfeldige er det på tide å se på noen nettverk som ikke er tilfeldige, men heller ikke naturlige. Vi starter med et datanettverk, der er objektene er datamaskiner eller rutere og forbindelsene er kommunikasjonskabler mellom dem. Dette er et planlagt nettverk, det betyr at egenskapene er bestemt på forhånd og at det så konstrueres et nettverk som har de gitte egenskapene. Datanettverk skal for eksempel være sammenhengende slik at alle datamaskinene kan kommunisere med hverandre, det skal være lett å sende meldingene til riktig mottaker. Største avstand mellom to maskiner skal ikke være for stor (gjelder bare store nettverk) for å minske tiden det tar å sende meldinger. Ikke minst skal kostnadene ved å bygge nettverket minimeres.

Datanettverk er ikke mulig å modellere med en tilfeldig modell. Hvor en ny kobling plasseres avhenger av helheten til nettverket og er ikke tilfeldig. En ting som er typisk for datanettverk er at dersom en forbindelse blir ødelagt så letes feilen opp og forbindelsen gjenopprettes, dette gjelder for mange planlagte nettverk. I et naturlig nettverk så finnes det ikke den samme sentrale styringen, hvis en forbindelse mellom to objekt blir ødelagt må et av de to objektene gjenopprette den. Siden hvert objekt bestemmer for seg selv er det ikke sikkert at en ødelagt kobling gjenopprettes. Det er mange egenskaper som vil være vanskelig å opprettholde i et naturlig nettverk, for eksempel er det vanskelig å ha få sykler uten å vite hvor mange sykler det finnes.

Veinettverk, nettverk av rørledninger og nettverket det objektene er flyplasser og forbindelsene er faste flyforbindelser er andre eksempler på planlagte nettverk.

2.4 Webgrafen

Webgrafen er grafen der hver internettside er en node og det er en rettet kant fra en side a til en side b hvis og bare hvis det er en lenke på side a som fører deg til side b. Det finnes per idag flere milliarder av websider. Siden all informasjon om webgrafen er lagret elektronisk er det mye lettere å samle informasjon om webgrafen enn for eksempel om et nettverk av vennskap. Lenker er lette å beskrive, enten er de til stede eller så er de ikke til stede. Det er ikke så mange nyanser og variasjoner i webgrafen som i et vennskapsnettverk. Likevel er det ikke lett å få oversikt over webgrafen, den er så stor at vi ikke har kontroll på alt og den endrer seg veldig fort.

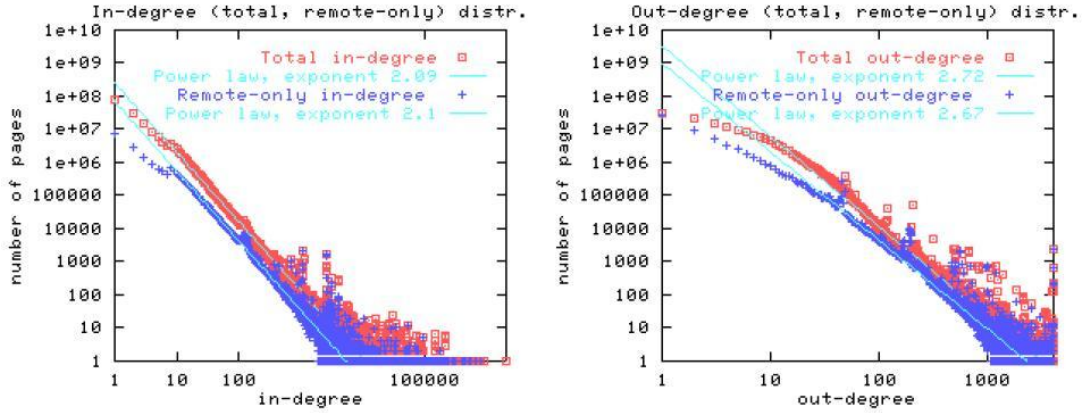
2.4.1 Samle informasjon om webgrafen

Mye resurser brukes på å holde oversikt over webgrafen, søkemotorer som yahoo og google lagrer store mengder informasjon for at brukerne skal finne det de leter etter så fort som mulig. For å samle informasjon er det programmer som søker ("crawler") igjennom sider ved å starte med en tilfeldig side og følge alle lenkene ut fra denne siden ved et bredde først søk. Men mesteparten av informasjonen holdes hemmelig for konkurrentene slik at forskningsmiljøene heller ikke får tilgang. Vi baserer observasjonene her på en artikkel av Broder et al [4] i 2000.

2.4.2 Webgrafen er ikke tilfeldig

Den informasjon som er tilgjengelig om webgrafen viser tydelig at webgrafen ikke kan beskrives av Erdős-Rényi modellen. Det tydeligste er kanskje fordelingen av gradtall. Mens tilfeldige grafer har gradtall som følger en binomisk fordeling, følger gradtallene i webgrafen en potens lov (power law) fordeling.

Litt forenklet betyr en potens lov fordeling av gradtallene at sannsynligheten for at en side har d lenker til andre sider er proporsjonal med $c * d^a$. Siden det er snakk om sannsynligheter og en sannsynlighet aldri kan være større enn 1 må a være negativ. c er en konstant som må justeres slik at sannsynligheten totalt blir 1. Eksponenten a er den viktigste karakterestikken ved en potens lov og er derfor ofte den eneste som blir oppgitt. For webgrafen



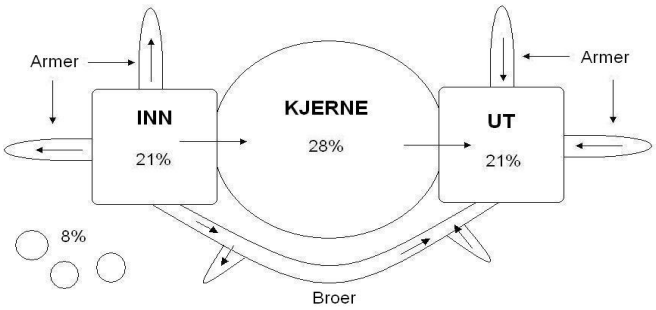
Figur 1: Observerte verdier for antall lenker fra og til en webside fra et søk på webgrafen i 1999 [4].

er eksponenten $a \approx -2.1$ for antall lenker inn til en side og $a \approx -2.7$ for antall lenker ut fra en side, noe som er bekreftet ved flere eksperimenter.

Figur 1 viser fordelingen av antall lenker inn til en side og antall lenker ut fra en side, funnet i et søk i 1999 [4]. Det er valgt å la begge aksene være logaritmiske. Funksjonen som er tegnet inn er en potens lov $f(x) = c * x^a$, tar vi logaritmen på begge sider får vi: $log(f(x)) = log(c) + a * log(x))$. Grunnen til at aksene er logaritmiske er at en potens lov fordeling da blir en rett linje.

En binomisk fordeling vil ikke bli en rett linje i et plot der begge aksene er logaritmiske. Vi kan heller ikke få en funksjon som ligner plotene i Figur 1, derfor kan ikke Webgrafen modelleres av Erdős-Rényi modellen. Som jeg nevnte i avsnittet om tilfeldige grafer er ikke dette nok til å konkludere om den er tilfeldig eller ikke. I en graf generert av Erdős-Rényi modellen har alle delgrafer samme struktur. Slik er det ikke i alle tilfeldige modeller. I webgrafen er det ikke slik, der avhenger sannsynligheten for en lenke sterkt av typen sider den lenker mellom, for eksempel språk og tema.

På mange internettsider er det ingen lenker til andre sider. Disse sidene inneholder gjerne informasjon eller forklaring på noe slik at mange andre sider lenker til disse sidene når denne forklaringen trengs. Derfor er det ikke overraskende at, fra mange sider kan ikke resten av webgrafen nåes. Det er også mange sider som ingen andre sider lenker til, derfor kan ikke disse sidene nåes fra resten av webgrafen. Hvis vi derimot ser på webgrafen som en urettet graf så er det nok å enten ha en lenke til siden eller lenke til en



Figur 2: Oversikt over hvordan webgrafen henger sammen. Armene som henger ut fra INN og UT pluss broene som går mellom INN og UT utgjør de resterende 22% av sidene.

side, derfor der det en større del som er sammenhengende i den urettede webgrafen. Det er klart at det kan finnes nyopprettede sider som ingen har lenket til ennå og som ikke har lenker til noen sider ennå. Det finnes også små interne nettområder der websidene kun har lenker mellom hverandre og er ikke ment å være koblet til Internett. Det er også mange hjemmesider som ikke har lenker ut og hvis det er lenker inn kommer de oftest fra andre hjemmesider som heller ikke har mange lenker. I undersøkelsen til Broder et al [4] var 92% av websidene en eneste sammenhengende komponent i den urettede webgrafen, når vi ser på webgrafen som en rettet graf blir det litt mer komplekst.

Figur 2 viser strukturen Broder et al [4] fant for den rettede webgrafen.

- KJERNE er den største sterkt sammenhengende komponenten, der kan alle par av sider nå hverandre ved å følge en kjede av lenker. Dette er kjernen av Internett og de fleste store nettsteder hører til i denne.
- UT er alle sider som kan nåes fra sider i KJERNE ved å følge en kjede av lenker, men der det ikke finnes stier som leder tilbake til KJERNE.
- INN er alle sider som har en kjede av lenker som fører til KJERNE.

I tillegg kommer en del sider som ikke kan nå KJERNE og som ikke kan nåes fra KJERNE, det er 4 typer slike sider.

1. de som kan nå UT og kan nåes fra INN
2. de som kan nå UT men ikke kan nåes fra INN
3. de som ikke kan nå UT men kan nåes fra INN

4. de som ikke kan nå UT og ikke kan nåes fra INN

Punkt nummer 1 over er broer mellom INN og UT, nummer 2 og 3 er inneholdt i armene som henger ut mens nummer 4 er enten i armene som henger ut eller sider som ikke er koblet til Internett. Fra figuren ser vi at det er mange par (a, b) av sider der det ikke gå an å følge en kjede av lenker fra side a til side b. Kun 25-45% av alle par av sider er forbunnet. Spørsmålet om hvilke mindre deler av webgrafen gir den samme prosentvise fordelingen, er uvisst.

Vi kan dele websidene inn i grupper som for eksempel sider skrevet på finsk og sider skrevet på norsk. Norsk og finsk er omtrent like store og viktige språk og derfor burde de samme dynamiske reglene gjelde for internettsider på begge språk. Hvis fordelingen av sider i de 4 gruppene i Figur 2 er en struktur som oppstår på grunn av dynamikken i webgrafen bør fordelingen være tilnærmet lik for de to språkene. Problemet med å sjekke en slik struktur er å finne ut hva som er de riktige gruppene å dele websider inn i. Hvis internettsidene kan deles inn i grupper som har den samme strukturen er dette nyttig informasjon som kan føre til bedre søkealgoritmer for internettsider.

2.4.3 Webgrafen er ikke planlagt

Det er så mange sider på Internett at ingen har oversikt over alle sidene, men søkemotorer gjør det mulig å finne de fleste sidene. Derfor kan man likevel få forholdsvis god oversikt over deler av webgrafen. I tillegg finnes det webportaler som leter opp de beste sidene, gjerne innen et bestemt tema, og samler lenker til alle disse sidene på et sted. Slike sider påvirker de som lager internettsider til å lage lenker til bestemte sider, men de kan ikke bestemme at en side skal lenke til en annen side. I et planlagt nettverk vil det være en sentral styring som kan bestemme over alle objektene, på Internett er det ingen slik styring. Innenfor hver server kan eieren av serveren bestemme mye, og noen firmaer eier mange servere. Derfor er det små deler av webgrafen som er planlagt, men som helhet er webgrafen ikke planlagt.

Det viktigste for en søkemotor er å finne de mest relevante sidene for et søkeord. For å avgjøre dette er teksten på siden og webgrafen eneste informasjon. Teksten bestemmes den enkelte webmaster og kan derfor manipuleres slik at søkemotorer tror en side er mer relevant for søket enn den egentlig er. Webgrafen er vanskeligere å manipulere, derfor er dette en viktig kilde for å vurdere hvor relevant en side er. Noen sider har mange lenker til andre sider, dette er ofte webportaler som samler lenker til relevante sider for at

brukere lettere skal finne informasjon. Sider som blir lenket til fra slike portaler er oftest gode og nyttige sider. Derfor regnes antall portaler som peker på en side som et godt mål for sidens kvalitet. Et firma som eier flere servere kan lage slike sider som lenker til mange andre sider og samtidig har en lenke til firmaets hjemmeside. Dette lurer søkemotorene til å tro at firmaets hjemmeside er viktigere enn den egentlig er.

Selv om vi ikke har funnet et endelig svar på hvor mye som er tilfeldighet i webgrafen er det stor grunn til å tro at den ikke er tilfeldig. Den er også for stor og uoversiktlig til å være planlagt. At webgrafen er et naturlig nettverk er nesten en forutsetning for at dette feltet skal fortsette sin raske utvikling, siden den er det nettverket som får informatikere til å interessere seg for naturlige nettverk. Derfor er vi et steg nærmere en definisjon av naturlige nettverk, nemlig at definisjonen må inkludere webgrafen. Naturlige nettverk er ikke tilfeldige nettverk, men kan inneholde stor grad av tilfeldighet, de er heller ikke planlagt, men kan inneholde mindre deler som er planlagt.

3 Strukturell balanse

En stor klasse av naturlige nettverk er de som kalles sosiale nettverk. Sosiale nettverk kan f.eks. beskrive forhold mellom personer. Disse forholdene kan være positive eller negative. Fordelingen av positive og negative forhold danner en struktur som vi kan gjenkjenne. For å modellere dette kan det brukes en graf der det er en node for hver person og en kant merket + for hvert par av venner og merket − for hvert par av fiender. Hvis det ikke er noen kant betyr det at de er nøytrale eller ikke har noe forhold til hverandre.

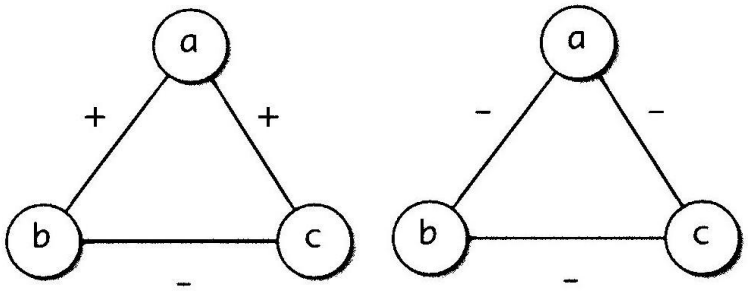
En person som har to venner, som igjen er fiender seg imellom, vil ofte prøve å mekle slik at hans to venner slutter fred. Hvis dette ikke går velger kanskje personen en side og gir opp det ene vennskapet for å styrke det andre. Hvis tre personer alle er fiender vil ofte to gå sammen mot en. Slike lokale egenskaper får stor innvirkning på helheten til nettverket, noe som fører til at personer har en tendens til å stå sammen mot en felles fiende. Derfor kan ofte en gruppe personer deles inn i to delgrupper (venner og fiender) slik at alle i samme gruppe er venner, mens alle i motsatt gruppe er fiender. Dette kalte Heider [10] for strukturell balanse. Siden mange nettverk ikke er strukturelt balansert ønsker vi å finne ut hvor ubalansert et nettverk er.

I dette kapitlet beskrives først strukturell balanse og en algoritme for å gjenkjenne strukturell balanse. Jeg viser også en alternativ måte å definere strukturell balanse. I andre del diskuteres en måte for å finne ut hvor ubalansert et nettverk er. Jeg ser da på 4 forskjellige indekser og vurderer kompleksiteten for å finne disse. Dette gjøres for å vise hva informatikk har å bidra med i dette feltet og for å vise at det ikke alltid er lett å bli enig om hva som er den beste definisjonen.

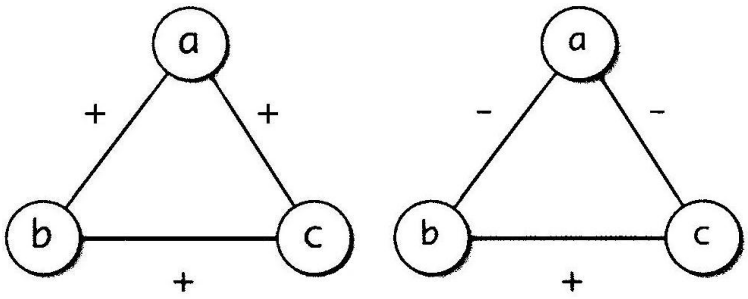
3.1 Definisjon av strukturell balanse

Når to fiender har en felles venn vil ofte denne felles vennen prøve å få de to til å bli venner, hvis dette ikke lykkes kan han bli nødt til å velge en av vennene fremfor den andre. Slik kan det argumenteres for at noen situasjoner er ustabile mens andre ikke er det. Ifølge Heider [10] er det 2 ustabile situasjoner mellom 3 personer, se Figur 3.

De ustabile situasjonene vil ha større sjanse for å endre seg enn andre situasjoner. Og når de endrer seg blir de en av de stabile situasjonene. Det er kun to stabile situasjonene med 3 personer der ingen par av personer er nøytrale, se Figur 4.



Figur 3: De to ubalanserte situasjonen som kan finnes.



Figur 4: De to stabile situasjonene der ingen par er nøytrale.

I tillegg er alle situasjonene der minst et par er nøytralt stabile situasjoner. Selv om nøytrale forhold er en lettvint måte å skape stabilitet er det i virkelige nettverk oftest mange tripler av personer der ingen par er nøytrale. Strukturell balanse ble senere generalisert av Cartwright og Harary [9], [5], det er denne definisjonen jeg bruker.

Definisjon 3.1. *(Strukturell balanse) [9], [5]*
Gitt en graf $G = (V, E)$ og en merking av kantene med $\{+, -\}$ (pluss og minus). G har strukturell balanse hvis og bare hvis det finnes en partisjonering av V i to partisjoner slik at $\forall (u, v) \in E$ er en av følgende to punkt oppfylt:

- *(u, v) er merket $+$ og u, v er i samme partisjon.*
- *(u, v) er merket $-$ og u, v er i forskjellige partisjoner.*

Når det er strukturell balanse finnes det altså ingen tilfeller av de ustabile situasjonene (se Figur 3 og 4, ingen av de ustabile situasjonene kan deles inn i to grupper slik at kantene oppfyller kravene for strukturell balanse mens begge to stabile kan).

Lemma 3.1. *La $G = (V, E)$ være en graf der kantene er merket med pluss eller minus. G har strukturell balanse $\Rightarrow$ alle delgrafer av G har strukturell balanse.*

Bevis. Siden G har strukturell balanse kan nodene deles inn i to delmengder L og R slik at disse oppfyller kravene i Definisjon 3.1. Når en kant fjernes opprettholdes strukturell balanse ved de samme delmengdene. Kravene i definisjonen legger kun restriksjoner på kantene så om vi fjerner en node opprettholdes også strukturell balanse. □

Lemma 3.2. *En graf G har strukturell balanse $\Leftrightarrow$ alle sammenhengende komponenter i G har strukturell balanse.*

Bevis. ($\Rightarrow$) Hvis G har strukturell balanse så har alle delgrafer av G strukturell balanse fra Lemma 3.1. Derfor har alle komponentene strukturell balanse.
($\Leftarrow$) Hver sammenhengende komponent c_k kan deles opp i to delmengder L_k og R_k slik at kantene oppfyller kravene i Definisjon 3.1. La $L = L_1 \cup L_2 ... \cup L_c$ og $R = R_1 \cup R_2 ... \cup R_c$ der c er antall sammenhengende komponenter i G. Nå oppfyller L og R alle kravene i Definisjon 3.1. □

Nå viser vi en algoritme som avgjør om en graf har strukturell balanse. Algoritmen ligner veldig på den som avgjør om en graf er bipartitt, siden kantene merket minus i en graf med strukturell balanse induserer en bipartitt graf.

Algoritme 3.3. *(Gjenkjenning av strukturell balanse)*

Inndata: en urettet graf $G = (V, E)$
Utdata: en partisjonering L, R av V
Lag lister L, R
Lag liste T
while** G eller T inneholder minst en node som ikke er i $L \cup R$ **do
  if** T inneholder en node u **then
    legg u til L
    if** ikke grafen indusert av $L \cup R$ har strukturell balanse **then
      flytt u fra L til R
    end if
    if** ikke grafen indusert av $L \cup R$ har strukturell balanse **then
      returner NEI.
    end if
  else
    finn en node u i G som ikke er i L eller R

legg u til i L
end if
for *alle noder v i G som er nabo av u* **do**
legg v til i T
end for
end while
returner L, R

Teorem 3.4. *Algoritme 3.3 gjenkjenner om en graf har strukturell balanse i lineær tid.*

Bevis. Hvis grafen har flere sammenhengende komponenter plasseres alle nodene i en komponent før første node i neste. Av Lemma 3.2 vet vi at det er nok å vise at algoritmen er korrekt for en sammenhengende graf. Den første noden plasseres i L og en graf på en node har strukturell balanse. Dette kan ikke være en feil start for å plassere den i R ville bare ført til at R kan byttes med L og noden er i L igjen. Anta at k noder er plassert. Inntil siste noden i en komponent er plassert inneholder T minst en node u siden grafen er sammenhengende. Siden u er i T må en av naboene til u allerede være plassert, la oss kalle naboen v. Hvis (u, v) er merket pluss må u plasseres i samme mengde som v, hvis den er merket minus må u plasseres i motsatt mengde av v. Uansett er det en unik måte å plassere nodene på. Algoritmen prøver begge, og hvis den ikke finner en måte betyr det at grafen ikke er strukturelt balansert. Når algoritmen skal teste om grafen indusert av $L \cup R$ har strukturell balanse trenger den bare å sjekke de nye kantene siden alle andre kanter er blitt sjekket før.

Hver node plukkes ut av G en gang og puttes i T. Så flyttes de fra T til L eller R. Dette tar tid $O(n)$. For hver node som fjernes fra T gåes det igjennom nabolisten til noden. Siden alle nodene er i T en gang tar dette tid $O(m)$. Kjøretiden er derfor $O(m + n)$. □

I begynnelsen snakket vi om stabile og ustabile situasjoner, og nå skal vi prøve å vise en sammenheng som får strukturell balanse til å henge bedre sammen med ideen om disse.

Lemma 3.5. *En graf G har strukturell balanse $\Leftrightarrow$ nodene til G kan partisjoneres i to delmengder slik at nøyaktig de stiene med begge endepunktene i samme partisjon har et jevnt antall kanter merket minus.*

Bevis. ($\Rightarrow$) Hvis vi traverserer en sti bytter vi delmengde hver gang vi traverserer en kant merket minus. Etter et jevnt antall ganger er vi tilbake der vi startet, mens etter et odde antall ganger er vi på motsatt side.
($\Leftarrow$) En kant merket minus er en sti med odde antall kanter merket minus,

mens en kant merket pluss er en sti med jevnt antall kanter merket minus. Alle kanter merket minus må ha endepunktene i forskjellige delmengder, mens kanter merket pluss må ha begge endepunktene i samme delmengde. Derfor har grafen strukturell balanse. □

Teorem 3.6. *[9], [5] En graf G har strukturell balanse ⇔ alle sykler har et jevnt antall kanter merket minus.*

Bevis. (⇒) En sykel er en sti som starter og ender i samme node og det følger dermed av Lemma 3.5.
(⇐) Av Lemma 3.2 vet vi at det er nok å vise teoremet for sammenhengende grafer. La $\sigma = v_1, v_2, ...v_n$ være en ordning av nodene slik at grafen G_k indusert av de første k nodene i σ er sammenhengende.

Nå gjør vi induksjon over k. En graf på en node har strukturell balanse, derfor holder det for $k = 1$.

Anta at G_k har strukturell balanse, i følge Lemma 3.5 kan nodene partisjoneres i to delmengder L og R kun stiene med begge endepunktene i samme partisjon har et jevnt antall kanter merket minus og alle stier som har endepunktene i forskjellige delmengder har et odde antall kanter merket minus. Siden G_{k+1} er sammenhengende vil v_{k+1} ha minst en nabo.

Hvis v_{k+1} kun har en nabo v_i: Hvis (v_i, v_{k+1}) er merket minus plasseres v_{k+1} i motsatt delmengde av v_i, ellers i samme. Det er altså kun 1 ny kant som blir lagt til og den blir lagt til slik at den oppfyller kravene i definisjon refdefSB.

Ellers har v_{k+1} minst to naboer v_i og v_j. Siden G_k er sammenhengende finnes det en sti fra v_i til v_j. Sammen med kantene (v_i, v_{k+1}) og (v_j, v_{k+1}) og stien danner en sykel. Hvis kantene (v_i, v_{k+1}) og (v_j, v_{k+1}) er merket:

- $-, -$ må stien fra fra v_i til v_j ha et partall antall kanter merket $-$ og derfor er v_i og v_j i samme delmengde.
- $+, +$ må stien fra fra v_i til v_j ha et partall antall kanter merket $-$ og derfor er v_i og v_j i samme delmengde.
- $+, -$ må stien fra fra v_i til v_j ha et odde antall kanter merket $-$ og derfor er v_i og v_j i motsatte delmengder.

Siden naboer av v_{k+1} som er forbundet med kanter merket pluss er i forskjellig delmengde fra de forbundet av kanter merket minus kan vi plassere v_{k+1} slik at alle kantene merket minus går til motsatt partisjon og alle kantene merket pluss går til samme partisjon. Det betyr at G_{k+1} har strukturell balanse. □

Merk at begge de to ustabile situasjonene i Figur 3 er sykler med odde antall kanter merket minus, mens begge de to stabile situasjonene i Figur 4 har jevnt antall kanter merket minus.

3.2 Mål for strukturell balanse

I de fleste større sosiale nettverk er det ikke strukturell balanse, siden en eneste person kan forstyrre balansen. Vi ønsker å beregne en indeks for strukturell balanse som sier hvor ubalansert et nettverk er. For å bestemme seg for hvilken indeks som er best å bruke, er det 2 punkt som må vurderes:

1. Indeksen må gjenspeile hvor nær et nettverk er å ha strukturell balanse.
2. Det må være praktisk mulig å bestemme eller i det minste tilnærme en verdi for indeksen.

Det første punktet er ikke klart definert. Selv om strukturell balanse er definert sier det ingenting om hvilke grafer som er nær å ha strukturell balanse. Før det er enighet om hva som skal måles er det lite informatikere kan gjøre. Men når en indeks dukker opp trengs det en algoritme for å måle den.

For å vise hva informatikere kan gjøre og samtidig illustrere at det er vanskelig å bli enig om et slikt mål skal vi vurdere 4 forslag.

Siden Definisjon 3.1 stiller krav til hver kant, er det naturlig å telle antall kanter som ikke oppfyller noen av punktene i definisjonen.

Definisjon 3.2. *Balanseindeks 1 for en graf er minimum antall kanter som må fjernes slik at grafen oppnår strukturell balanse.*

Teorem 3.7. *Å avgjøre om Balanseindeks 1 $\leq k$ for en graf G er NP-komplett.*

Bevis. Kan vises ved reduksjon fra det NP-komplette problemet største kutt (Max-cut). La G være en graf med m kanter og la H være en kopi av G, der alle kantene er merket minus.
G har et kutt av størrelse $m - k \Leftrightarrow H$ har Balanseindeks $1 \leq k$.
($\Rightarrow$) Det er k kanter i G som ikke er del av kuttet, la H' være grafen som gjenstår når de k tilsvarende kantene fjernes fra H. Nodene i H' deles inn i to delmengder slik at alle kantene i H' har et endepunkt i hver delmengde. Siden alle kantene i H er merket minus har H' strukturell balanse.
($\Leftarrow$) Det finnes k kanter som kan fjernes fra H og nodene kan deles inn i to delmengder slik at de $m - k$ kantene som er igjen har et endepunkt i hver delmengde. De tilsvarende $m - k$ kantene i G er et kutt.
Siden største kutt er NP-komplett er Balanseindeks 1 NP-komplett. □

Teorem 3.8. *Å avgjøre om Balanseindeks 1 for grafen $G = (V, E)$ er lik k er FPT med k som parameter.*

Bevis. Vi skal omforme problemet til å finne største bipartitte delgraf. Lag en ny graf $H = (V_H, E_H)$ der all kanter i G merket minus også er kanter i H, mens alle kanter i G merket pluss blir erstattet av en sti med lengde 2.
G har Balanseindeks $1 \leq k \Leftrightarrow \exists K \subset E_H$ der $|K| \leq k$ slik at $H \setminus K$ er en bipartitt graf.
($\Rightarrow$) Hvis det finnes k kanter i G som gir G' hvis de fjernes, slik at G' har strukturell balanse. Kan vi hvis kanten er merket minus fjerne tilsvarende kant i H og hvis kanten er merket pluss fjerne en av de to kantene vi laget i H og får da en ny graf H'. Det er da k kanter som er fjernet fra H. I G' kan nodene deles inn i to sider slik at alle kantene merket pluss har da begge endepunktene på samme side og alle kanter merket minus har et endepunkt på hver side. For å vise at H' er bipartitt deles H inn i to sider på samme måte som G'. I H' er alle kanter merket pluss i G' erstattet av en sti av lengde 2, den nye noden plasseres da på motsatt side av sine naboer, som er på samme side, siden G' har strukturell balanse. Begge de 2 kantene i stien i H' har da et endepunkt på hver side. Hvis en kant merket pluss i G ikke er i G' er kun en av de to kantene i H', denne kanten plasseres med et endepunkt på hver side (alltid mulig siden ene endepunktet er et løv). Kantene merket minus i G' har et endepunkt på hver side, derfor har alle tilsvarende kanter i H' et endepunkt på hver side. Så alle kantene i H' har et endepunkt på hver side. H' er derfor bipartitt.
($\Leftarrow$) La H' være en bipartitt graf som gjenstår etter at k kanter er fjernet fra H. La G' være grafen som gjenstår etter at følgende k kanter er fjernet fra G: Kanter i G merket minus som tilsvarer en kant i H som ikke er i H' fjernes. Kanter i G merket pluss som tilsvarer en sti i H som ikke er i H fjernes. Nodene i G' deles inn i samme to delmengder som viser at H' er bipartitt. Alle kanter i G' merket minus tilsvarer en kant i H' og har derfor et endepunkt i hver delmengde. Alle kanter i G' merket pluss tilsvarer en sti av lengde 2 i H'. Endepunktene til en sti av lengde 2 i en bipartitt graf ligger i samme delmengde, derfor har alle kanter merket pluss i G' begge endepunktene i samme delmengde.

Å bestemme om k kanter kan fjernes fra en graf slik at en grafen blir bipartitt kalles Edge Bipartization, se Guo et al. [8] for en $O(2^k * km^2)$ algoritme, der m er totalt antall kanter i grafen. I beviset for Teorem 3.8 har vi lagt til noen noder og kanter, men aldri mer enn dobbelt av den originale grafen derfor holder samme kjøretid for Balanseindeks 1. □

En naturlig vri av forrige definisjon er å bytte kanter med noder.

Definisjon 3.3. *Balanseindeks 2 for en graf er minimum antall noder som må fjernes slik at grafen oppnår strukturell balanse.*

Teorem 3.9. *Å finne Balanseindeks 2 for en graf G er NP-komplett.*

Bevis. Kan vises ved reduksjon fra største induserte bipartitte delgraf. La G være en graf med n noder og la H være en kopi av G, der alle kantene er merket minus.
Det finnes k noder slik at G blir bipartitt når disse fjernes $\Leftrightarrow$ H har Balanseindeks $2 \leq k$
($\Rightarrow$) Fjernes de tilsvarende k nodene fra H betyr det at nodene i H kan deles inn i to delmengder slik at alle kantene har et endepunkt i hver delmengde, siden alle kantene i H er merket minus betyr det at Balanseindeks 1 for H er mindre enn eller lik k.
($\Leftarrow$) Fjernes de tilsvarende k nodene fra G kan nodene i G deles inn i to delmengder slik at alle kanter som tilsvarer en kant merket minus i H har et endepunkt i hver delmengde, siden alle kanter er merket minus i H betyr det at G er bipartitt.
Siden største induserte bipartitte delgraf (Graph Bipartization) er NP-komplett er Balanseindeks 2 NP-komplett. □

Det finnes en enkel $O(n^k)$ algoritme der k er antall noder som må fjernes for å gjøre grafen bipartitt. Antall noder som må fjernes er mindre enn antall kanter fordi å fjerne en av endenodene til en kant også fjerner kanten, derfor er k mindre for Balanseindeks 2 enn for Balanseindeks 1, likevel er $O(n^k)$ for stor kjøretid, vi skal prøve å finne en FPT algoritme for denne også.

Teorem 3.10. *Å avgjøre om Balanseindeks 2 på grafen $G = (V, E)$ er lik k, er FPT med k som parameter.*

Bevis. Samme konstruksjon som for Balanseindeks 1 kan brukes. Lag en ny graf $H = (V_H, E_H)$ der all kanter i G merket minus også er kanter i H, mens alle kanter i G merket pluss blir erstattet av en sti med lengde 2. Men Balanseindeks 2 fjerner noder, derfor må argumentene gjøres om igjen.
G har Balanseindeks $1 \leq k \Leftrightarrow \exists K \subset E_H$ der $|K| \leq k$ slik at $H \setminus K$ er en bipartitt graf.
($\Rightarrow$) Hvis det finnes k noder i G som gir G' hvis de fjernes, slik at G' har strukturell balanse. Kan vi fjerne tilsvarende noder i H og får da en ny graf H'. I G' kan noden deles inn i to sider slik at alle kantene merket pluss har da begge endepunktene på samme side og alle kanter merket minus har et endepunkt på hver side. For å vise at H' er bipartitt deles H inn i de samme to sidene som G'. I H' er alle kanter merket minus i G' erstattet av en sti av lengde 2, den nye noden plasseres da på motsatt side av sine naboer, som er på samme side siden G' har strukturell balanse, slik at begge de 2 kantene i stien i H' har et endepunkt på hver side. Hvis en kant merket pluss i G

ikke er i G' er kun en av de to kantene i H', denne kanten plasseres med et endepunkt på hver side (alltid mulig siden ene endepunktet er et løv). Kantene merket minus i G' har et endepunkt på hver side, derfor har alle tilsvarende kanter i H' et endepunkt på hver side. Så alle kantene i H' har et endepunkt på hver side. H' er derfor bipartitt.
($\Leftarrow$) Hvis grafen H' er bipartitt, så er det to typer noder som kan være fjernet fra H. Hvis noden har en tilsvarende node i G så fjernes samme node fra G, det er aldri nødvendig å fjerne noder som er en del av en sti av lengde 2 som har erstattet en kant i G merket pluss, vi kan fjerne en av naboene istedenfor. Å fjerne en av naboene gjør at en av kantene i stien blir værende, men noden på stien blir et løv så den kan alltid plasseres slik at kanten har et endepunkt på hver side. Det at alle noder som er fjernet fra H har en tilsvarende node i G. Alle kanter merket minus i G' har en tilsvarende kant i H' som har et endepunkt på hver side, derfor har alle kantene i G' merket minus et endepunkt på hver side. Alle kantene i G' merket pluss har en tilsvarende sti av lengde 2 i H' der begge kantene har et endepunkt på hver side, derfor har alle kanter merket pluss i G' begge endepunktene på samme side. Altså har G' strukturell balanse.

Å finne antall noder som må fjernes for å få en bipartitt graf kalles Graph Bipartization og kan løses i $O(3^k * mn)$ tid, se Falk Hüffner [11]. I beviset for Teorem 3.10 har lagt til noen noder og kanter, men aldri mer enn dobbelt av det den originale grafen hadde, derfor holder samme kjøretid for Balanseindeks 2. □

Det er vanskelig å avgjøre hvilke av indeks 1 og 2 som best måler strukturell balanse. Det er et spørsmål som samfunnsvitenskapen må svare på. Informatikk kan kun prøve å designe algoritmer som bestemmer indeksene så får andre bestemme om indeksene sier noe nyttig om nettverk. Begge algoritmene har FPT algoritmer, det som bestemmer om de er mulig å bruke i praksis er størrelsen på k. Det er antatt at k er liten for de fleste sosiale nettverk, og siden k alltid er mindre for Balanseindeks 2 er dette antagelig den som er lettest å beregne.

Definisjon 3.4. *La Balanseindeks 3 være* 1 *for en graf uten sykler. For alle andre grafer er Balanseindeks 3 forholdet mellom antall sykler med jevnt antall kanter merket minus og totalt antall sykler.*

Dette var den indeksen Tom Backer Johnsen ved Institutt for samfunnspsykologi ved UiB ønsket å beregne når jeg kom i kontakt med ham. Det er også en indeks som er godt omtalt i litteraturen.

Denne indeksen er litt annerledes. Det er ikke naturlig å lage et JA/NEI spørsmål ut av dette. Det er derimot naturlig å splitte indeksverdien opp i to

deler, den over og den under brøkstreken. Vi skal vise at det er vanskeligere å bestemme Balanseindeks 3 enn å finne totalt antall sykler.

Teorem 3.11. *Å finne Balanseindeks 3 for en graf G er ♯P-hardt.*

Bevis. Kan vises ved reduksjon fra det ♯P-harde problemet å telle antall sykler i en graf. La G være en graf, konstruer en graf H ved å lage en kopi av G og merk alle kantene med pluss. Legg så til 3 nye noder der alle er forbundet med hverandre, alle kantene mellom disse nodene merkes minus.
Antall sykler i $G = k \Leftrightarrow$ Balanseindeks 3 for $H = \frac{1}{k+1}$.
($\Rightarrow$) Siden det er k sykler i G vil det være $k+1$ sykler i H, vi la til en sykel av lengde 3. Dette er den eneste sykelen med odde antall kanter merket minus. Derfor blir Balanseindeks 3 for $H = \frac{1}{k+1}$. ($\Leftarrow$) Det er kun en sykel med odde antall kanter merket minus, derfor må totalt antall sykler i H være $k + 1$. Det betyr at det er k sykler i G.
Antall sykler er ♯P-komplett (kan vises ved reduksjon fra antall Hamiltonske sykler) derfor er Balanseindeks 3 ♯P-hard. □

Det er altså veldig lite håp for at vi kan bestemme Balanseindeks 3 nøyaktig. Det interessante med denne indeksen er at det finnes en god måte å finne en tilnærmet verdi. Algoritme 3.12 tilnærmer Balanseindeks 3 for en graf med høy sannsynlighet. Algoritmen prøver L tilfeldige ordninger. For hver ordning teller den opp hvor mange sykler av forskjellige lengder som følger ordningen og inneholder den valgte startnoden. Den holder også telling på hvor mange av syklene som har et jevnt antall kanter merket minus. Algoritmen teller alle sykler av forskjellig lengde hver for seg. Etter at alle sykler som følger en ordning er funnet må det summeres opp hvor mange sykler som er funnet men, siden korte sykler blir funnet i mange flere ordninger enn lange sykler, må sykler av forskjellig lengde bidra forskjellig til summen. For en sykel av lengde k legges $(k-1)!$ til summen for enten jevne eller odde sykler, fordi sannsynligheten for å finne en sykel av lengde k er omvendt proporsjonal med $(k-1)!$. Denne summen brukes til slutt når forholdet mellom jevne og totalt antall sykler skal regnes ut. Hvis L økes gir algoritmen et bedre resultat, men det betyr også lengre kjøretid.

Algoritme 3.12. *(Estimering av indeks 3)*

Inndata: Graf på n noder der kantene er merket '+' eller '-'; et heltall L
Utdata: en tilnærming av Balanseindeks 3
definer J[n][n], O[n][n]
definer sumJ og sumO
for** 1 to L **do
  finn en tilfeldig permutasjon r av nodene (som ikke er brukt før).

```
sett alle elementene i J og O til 0.
J[0][0] = 1;
for i = 0; i ≤ n; i++ do
  for j = 1; j < i; j++ do
    for k = 1; k ≤ i; k++ do
      if (r(j), r(i)) er '-' kant then
        J[i][k]+ = O[j][k − 1]
        O[i][k]+ = J[j][k − 1]
      end if
      if (r(j), r(i)) er '+' kant then
        J[i][k]+ = J[j][k − 1]
        O[i][k]+ = O[j][k − 1]
      end if
    end for
  end for
end for
for k = 1; k <= n; k++ do
  sumJ+ = J[n][k] ∗ (k − 1)!
  sumO+ = O[n][k] ∗ (k − 1)!
end for
end for
Return sumJ/(sumJ + sumO)
```

Hvis algoritmen prøver alle mulige ordninger blir hver sykel tellt nøyaktig $(n-1)!$ ganger, derfor får vi et nøyaktig svar hvis vi har kapasitet til å prøve alle $n!$ ordningene. Større verdi av L betyr at flere ordninger blir prøvd og bør derfor gi en bedre tilnærming. Jeg har ikke klart å regne ut nøyaktig hvor god denne tilnærmingen er, det er derfor et åpent problem.

Siden det kan finnes $O(2^n)$ sykler som følger en gitt ordning bruker tabellene våre $O(n^3)$ minne. For å finne en ordning som ikke er brukt før trengs en liste over de brukte ordningene, hvis dette en sortert liste kan en tilfeldig ordning finnes og legges til i $O(n + log(L)) \subseteq O(nlog(n))$ tid. For å finne antall sykler som følger en ordningen trengs $O(n^3)$ tid. Alt dette gjøres for hver permutasjon så kjøretiden blir $O(L * n^3)$.

Vi kan forvente en vilkårlig høy presisjon ved å bruke ekstra kjøretid. Slike algoritmer er ofte gode i praksis.

Siden personer som er nær hverandre (naboer eller naboer av naboer) påvirker hverandre mye mer enn de som er langt fra hverandre er det mye viktigere at korte sykler inneholder jevnt antall minuser enn at lange sykler gjør det. Dette reflekteres ikke av indeks 3.

Definisjon 3.5. *La Balanseindeks 4 være* 1 *for en graf uten sykler. For alle andre grafer er Balanseindeks 4 med parameter k lik* $\frac{J_k}{T_k}$ *der* J_k *er antall sykler med lengde høyst k med jevnt antall kanter merket minus og* T_k *er totalt antall sykler av lengde høyst k.*

I motsetning til Balanseindeks 3 finnes det en enkel algoritme som kjører på tid $O(n^k)$ som beregner Balanseindeks 4 for parameter k. Det gjør at denne er mer anvendt enn Balanseindeks 3. Algoritme 3.12 kan lett modifiseres til å tilnærme Balanseindeks 4.

Vi ser at det er mange utfordringer som må overkommes før vi har konkrete mål på naturlige nettverk. Strukturell balanse er en veldefinert egenskap, men få nettverk har denne egenskapen. Når vi prøver å finne et mål eller en måte å klassifisere nettverk bør det gjøres slik at vi får en bred enighet mellom samfunnsvitenskap og informatikk, men det er da mange faktorer det må tas hensyn til. Kompleksiteten er en viktig ting, og informatikere kan være til stor hjelp her, samtidig bør vi gi samfunnsvitenskapen litt fleksibilitet og valgmuligheter for eksempel slik jeg har gjort her ved å gi 4 indekser. Det er mange problemer i denne retningen som ikke er angrepet på denne måten ennå, og kan åpne for mye samarbeid og nye resultater.

4 Verden er liten

4.1 Introduksjon

Det er vanlig å si at "verden er liten", men er den det? Og hva mener vi med det? I dette kapitlet vil vi se på det som kalles "verden er liten"(small world) fenomenet, med fokus på et eksperiment utført i 1969 av Milgram [15]. Med verden mener vi et stort bekjentskapsnettverk. Det trenger ikke være alle mennesker i verden men kanskje et land eller et annet stort sammenhengende område. Vi skal også beskrive en modell som har samme egenskaper som Milgram fant.

Definisjon 4.1 (bekjentskapsnettverk)**.** *En graf der nodene er personer og der to noder er forbundet med en kant hvis og bare hvis personene kjenner hverandre på et personlig plan.*

Definisjon 4.1 bruker et begrep som "kjenner hverandre på et personlig plan" selv om det ikke er lett å finne en klar definisjon på slike ting.

Milgram gjorde et eksperiment i USA [15] i 1969 der han ville vise at verden var liten. Han undersøkte lengden på korte stier i bekjentskaps nettverket for USA. 296 tilfeldige avsendere fikk et brev til en person i Massachuset sammen med informasjon om personens yrke, bosted og litt personlige opplysninger. I brevet var det informasjon om å sende brevet videre til en bekjent, slik at brevet så raskt som mulig kom frem til mottaker. Håpet var at brevet skulle komme frem til denne mottakeren uten å måtte sendes for mange ganger. Eksperimentet viste at gjennomsnittlig måtte brevet sendes et sted mellom 5 og 6 ganger før det nådde mottaker hvis det i det hele tatt kom frem. Selv om "Six degrees of separation"[7] er blitt et anerkjent begrep brukte Milgram ikke dette begrepet. Det er en del unøyaktigheter forbundet med dette forsøket, mindre enn 1/3 av brevene kom frem og mottaker var en velutdannet person med mange kontakter. Dessuten var det lite kontroll på deltakernes bekjentskap, hva er et bekjentskap og ble brevene utelukkende sent mellom bekjente? Uansett, dette eksperimentet antyder to ting om bekjentskapsnettverk:

1 Det finnes korte kjeder av bekjentskap mellom to tilfeldige personer.

2 Personene klarer selv å finne korte kjeder med kun begrenset informasjon om mottakeren

4.2 Korte stier

Det er ikke helt klart hva som menes med korte kjeder av bekjentskap. En kjede er en sti i grafen som beskriver nettverket. Det som menes er at forventet lengde på korteste sti mellom to tilfeldige noder er $O(log(n)^c)$ for enhver konstant c. En komplett graf på n noder har avstand 1 mellom alle par av noder, men dette er ikke en god modell siden vi ønsker at hver node ha få naboer. Følgende teorem av Bollobás og de la Vega [3] viser en modell der forventet lengde på stiene er $O(log\ n)$, selv om hver node har et konstant antall naboer.

Teorem 4.1. *[3] La U være mengden av alle sammenhengende k-regulære grafer med n noder, der $k > 2$. Hvis vi velger uniformt fra U får vi med høy sannsynlighet en graf T der alle par av noder er forbundet ved en sti av lengde $O(log_k\ n)$*

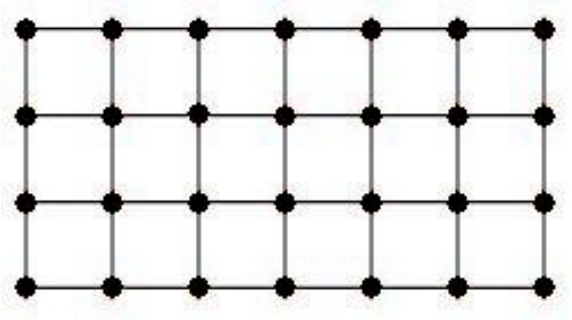
Dette er en veldig enkel modell som har stier av lengde $O(log\ n)$. Det er selvsagt ikke viktig at grafen er regulær, enkelte noder kan gjerne ha flere naboer. Grunnen til at vi må ha gradtall større en 2 er for at grafen med stor sannsynlighet skal bli sammenhengende. Poenget er at selv med $O(n)$ kanter kan vi finne en graf med stier av lengde $O(log(n))$. I Milgrams eksperiment fant de lengde 6, blant de ca 200 millioner menneskene som bodde i USA på den tiden. $log_{17}(2 * 10^8) \approx 6$, så en graf på 200 millioner noder med gradtall 17 kan tenkes å ha stier av lengde 6. Fra nå av kaller vi alle stier av lengde $O(log(n)^c)$ der c er en konstant for korte stier.

4.3 Finne korte kjeder i et nettverk

Selv om en slik modell har korte stier er det ikke sikkert den vil virke som modell for et bekjentskapsnettverk fordi det kan være umulig å finne stiene slik personene gjorde med brevene i Milgram's eksperiment.

Hvis du får et brev til en navngitt mottaker aner du ikke hvilken retning du skal sende brevet og ender opp med å send brevet til en tilfeldig bekjent. Det er da lite sannsynlig at brevet kommer frem selv etter hundrevis av sendinger. Hvis du derimot har en geografisk posisjon for mottaker kan du sende brevet til en bekjent av deg som bor nærmere og slik være rimelig sikker på at brevet kommer raskt frem. Denne observasjonen var starten på en ny måte for å lage modeller.

Watts og Strogatz [16] var pionerene og la frem en modell som gir personer denne informasjonen og dermed har mange av de egenskapene et naturlig



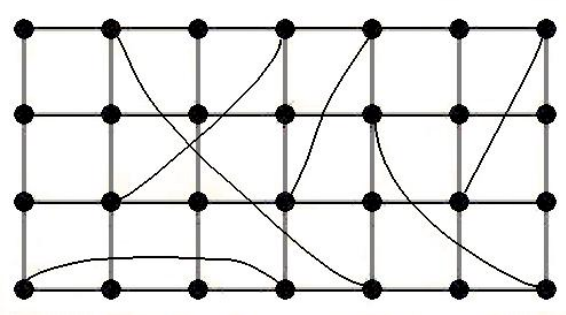
Figur 5: Et 4x7 rutenett

nettverk har. Selv om målet med modellen ikke var å forklare Milgrams eksperiment viste det seg at modellen var en god start på dette. Vi skal vise en variant av denne modellen, la oss kalle den rutenettmodellen. Selv om den modellen jeg legger frem ikke er helt lik den Watts og Strogatz la frem viser den de samme prinsippene. Modellen jeg legger frem er stort sett lik modellen til Jon Kleinberg [14].

Definisjon 4.2 (rutenett)**.** *På et euklidsk plan, la alle heltalls koordinatene være en node. La alle noder som har avstand 1 være forbundet av en kant. Nodene som har koordinater* (x, y) *slik at* $0 \leq x < m$ *og* $0 \leq y < l$ *induserer et* $m * l$ *rutenett. (Se figur 5)*

Med et $\sqrt{n} \times \sqrt{n}$ rutenett, la oss kalle det R har vi en basis for vår modell med n noder. Vi har også definert en avstandsfunksjon, den euklidske avstanden (korteste sti) i rutenettet. Det sikrer at to noder i avstand d fra hverandre er forbundet av en sti av lengde d som er lett å finne. Hvis personene nå kjenner koordinatene til mottaker kan de, ved alltid å sende meldingen til en som er nærmere i rutenettet, sørge for at meldingen kommer frem, selv om det kan bli en sti av lengde $\Theta(\sqrt{n})$.

Siden rutenettet ikke har korte stier mellom alle par av noder må noen ekstra kanter legges til for å sikre korte stier. Vi kan bruke den tilfeldige grafen T fra Teorem 4.1 (la rutenettet R og grafen T ha samme nodemengde med n noder). Ved å legge T oppå R sikrer vi at det finnes korte stier i modellen. Nå vet vi at det finnes korte stier (ved å følge kantene i T) og vi vet at personene selv kan finne en sti av lengde $O(\sqrt{n})$ (ved å følge kantene i R). Det neste spørsmålet er om personene selv kan finne korte stier i $T \cup R$? Personene kan ikke følge den korte stien som finnes i T siden de ikke vet hvor den går. Vi antar at alle personer kjenner posisjonen til sine naboer, uansett om det er en nabo i T eller R. De må også utnytte det at de vet posisjonen til mottaker. Følgende blir den mest naturlige algoritmen:



Figur 6: En indusert subgraf av et 12x12 rutenett R med en graf T lagt oppå.

Algoritme 4.2. *(Søk etter korte stier)*
while *meldingen ikke har nådd mottaker* ***do***
send meldingen til den av naboene i grafen $T \cup R$ som har kortest avstand til mottaker i grafen R
end while

I begynnelsen så vil algoritmen gå raskt nærmere ved å følge de tilfeldige kantene som går i riktig retning, men jo nærmere mottaker meldingen kommer jo mindre blir sannsynligheten for å finne en tilfeldig kant som fører nærmere mottaker. (Se figur 6 som viser hvordan et 12x12 rutenett kan se ut når meldingen er nær mottaker). Derfor vil dette ikke føre til en algoritme som finner stier av lengde $O(log(n))$.

Vi har fremdeles ikke en tilfredsstillende modell. Før vi går videre la oss innføre litt notasjon. Med R-avstand menes lengden på den korteste stien mellom to noder i rutenettet R. Med T-nabo (D-nabo) menes en node som er nabo i grafen T (D).

Problemet med modellen var at når meldingen hadde liten R-avstand til mottaker var det for få T-naboer som hadde kortere R-avstand til mottaker, derfor må meldingen mot slutten med stor sannsynlighet følge kantene i R. Kleinberg foreslo å la D være en graf med samme nodemengde som R der sannsynligheten for at et tilfeldig par (u, v) er forbundet med en kant er avhengig av R-avstanden mellom u og v. Ved å bytte ut T med D kan det bli lettere å finne en kant som fører nærmere mottaker når meldingen er nær mottaker. Kleinberg foreslo at sannsynligheten for at en kant er i D skulle være proporsjonal med $f(n)/d(u, v)^q$, der $d(u, v)$ er R-avstanden mellom u og v og $f(n)$ er en normaliseringsfunksjon for at alle noder skal ha et konstant antall naboer.

Hvis q settes til 0 blir kantene helt uniformt fordelt i D (ikke avhengig av avstanden) og har derfor egenskaper som ligner grafen T. Når q blir større blir det færre og færre lange kanter og når $q \to \infty$ blir det kun kanter som allerede finnes i R. Det må derfor være en verdi imellom 0 og ∞ som gir de korteste stiene.

Kleinberg gjorde en simulering på 400 millioner noder der han brukte Algoritme 4.2, som ga at den beste verdien for q var mellom 1.5 og 2.

Det er noe komplisert å regne ut hvilke verdier av q som er best, spesielt å finne nedre grenser på de. Men det viser seg at $q = 2$ er en god verdi. Kleinberg [13] viste at $q = 2$ er den eneste verdien som garanterer korte stier.

Lemma 4.3. *Grafen D med n noder har $\Theta(n)$ kanter når $q = 2$ og $f(n) = 1/log(n)$.*

Bevis. La v være en node i midten av R (siden nodene i midten har høyest forventet gradtall). Nodene som har R-avstand k fra v danner et kvadratt som er vridd 45 grader, det er akkurat $4 * k$ noder med R-avstand k (unntatt når avstanden til kanten av R er mindre enn k). For å finne forventet antall D-naboer N til noden v summerer vi sannsynligheten for en kant til v over alle nodene. Forventet antall D-naboer til v som har R-avstand k er derfor $4*k*1/log(n)*1/k^2 = 4/(k*log(n))$. Den største mulige avstand fra sentrum av et rutenett er $\sqrt{n}$. Summerer vi over alle verdier av k får vi.

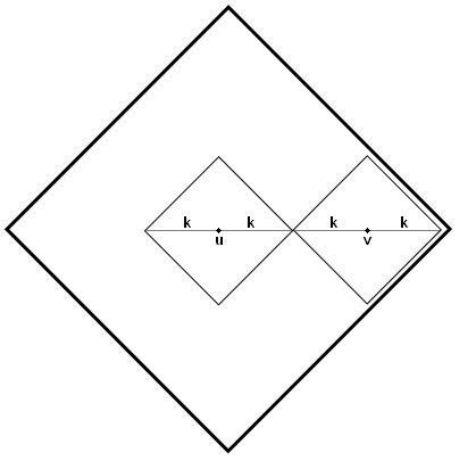
$$N \leq \sum_{k=1}^{\sqrt{n}} \frac{4}{log(n) * k} = \frac{4}{log(n)} * \sum_{k=1}^{\sqrt{n}} \frac{1}{k}$$

$1/k$ er den Harmoniske rekken. $\sum_{k=1}^{\sqrt{n}} 1/k = \Theta(\ln \sqrt{n}) = \Theta log(n)$, vi får derfor at: N er $O(1)$.

Nodene i hjørnene av R er de som har minst gradtall, derfor får vi en nedre grense for N hvis vi gjentar summasjonen over. Denne nedre grensen viser at N er $\Omega(1)$ og derfor kan vi konkludere at D-naboer er $\Theta(1)$ for alle noder. For å finne forventet antall kanter i D summerer vi forventet antall D-naboer og deler på 2. Dette gir at forventet antall kanter i D er $\Theta(n)$. □

Teorem 4.4. *[13] I grafen $D \cup R$ der en tilfeldig kant er i D med sannsynlighet $1/(log(n) * d(u,v)^2)$, vil Algoritme 4.2 finne stier av lengde $O(log(n)^2)$.*

Bevis. Ved å summere over antall D-naboer med R-avstand mellom k og $2*k$ får vi at antall D-naboer u av v slik at $k < d(u,v) < 2 * k$ er proporsjonalt med $1/log(n)$. Antall D-naboer med R-avstand mellom k og $2 * k$ er altså ikke avhengig av k.



Figur 7: Viser to noder u og v i et rutenett, de små kvadratene markerer nodene med avstand k fra henholdsvis u og v, mens det store kvadratet markerer nodene med avstand $3k$ fra u.

Anta at en melding skal sendes fra u til v i grafen $D \cup R$, og $d(u,v) = 2 * k$. Vi vil nå finne forventet antall steg for å halvere avstanden. For enhver node er forventet antall D-naboer som har R-avstand mellom k og $3 * k$ av størrelsesorden $\frac{1}{log(n)}$. Av alle noder x som har $k \leq d(u,x) \leq 3k$ så er sannsynligheten for at $d(x,v) \leq k$ tilnærmet 1/8 se figur 7 (det er plass til 8 små kvadrater i området mellom k og $3k$. Dermed er forventet antall kanter som halverer avstanden av størrelsesorden $\frac{1}{8*log(n)}$. Etter at algoritmen har besøkt $O(8*log(n))$ noder kan vi forvente at en nabo som halverer avstanden til mottaker er funnet. For å få beskjeden frem trenger vi å halvere avstanden $O(log(n))$ ganger. Algoritme 4.2 bruker derfor gjennomsnittlig $O(log(n)^2)$ steg på å sende meldingen til rett mottaker. □

Med denne algoritmen på denne modellen får vi en forventet lengde på stien Algoritme 4.2 finner mellom to tilfeldige noder på $O(log(n)^2)$. Dette viser at ikke bare finnes det korte stier, men personene selv kan også finne slike. I R har nodene maks 4 naboer som gir $\Theta(n)$ kanter i R, og grafen D har fra Lemma 4.3 $\Theta(n)$ kanter. Så med $\Theta(n)$ kanter i $D \cup R$ kan stier av lengde $\Theta(log(n)^2)$ finnes. Hvis personene har $O(log(n))$ bekjente kan stier av lengde $O(log(n))$ finnes.

5 FPT og trelignende grafer

Vi så under kapitlet Strukturell balanse at algoritmer for å gjenkjenne grafstrukturer kan være tidkrevende. Det er veldig mange grafproblemer som er NP-komplette. Noen ganger er det behov for en algoritme for et problem som er NP-komplett samtidig som grafen er så stor at en rå-kraft algoritme blir for langsom. Vi så eksempler på dette i kapittelet om Strukturell balanse, hvor vi fant FPT algoritmer som var mye raskere enn rå-kraft.

Første alternativ for å angripe et slikt problem på, er tilnærmingsalgoritmer. Det er algoritmer som gir tilnærmet riktig svar. Tilnærmingsalgoritmer virker ikke på problemer med JA/NEI svar, siden det bare er 2 mulige svar finnes det ingen svar som er tilnærmet riktig. Er det derimot et problem som spør om antall løsninger eller om svaret er et tall virker tilnærming mye bedre. Med tilnærmet mener vi vanligvis at svaret ikke skal avvike med mer enn en faktor $(1 \pm \varepsilon)$ fra en optimal løsning uansett inndata. Det ikke alltid lett å tilnærme problemer som spør etter et tall heller. Er spørsmålet: finn antall hamiltonske sykler, så er det ofte umulig å tilnærme svaret. Hvis det finnes mange hamiltonske sykler kan vi finne noen av disse og dermed ha et tilnærmet svar, men finnes det veldig få, kan vi lete etter sykler uten å finne noen og algoritmen vil svare 0 som kan være ganske forskjellig fra svaret. Tilnærmingsalgoritmer virker derfor best på problemer som har mange løsninger.

Andre alternativ er å utnytte ekstra informasjon vi har om inndata. Hvis vi vet at inndata er et naturlig nettverk, kan det godt hende at problemet er NP-komplett for generelle grafer mens for de fleste naturlige nettverk lar det seg løse. Vi vet for eksempel at de fleste naturlige nettverk har korte stier mellom nesten alle par av noder, kan dette brukes til noe? Dette er spørsmål som stilles innen fagfeltet kalt parametrisert kompleksitet og målet er FPT algoritmer.

Parametrisert kompleksitet er når programmet i tillegg til inndata av lengde n får en parameter k som beskriver en egenskap ved inndata. Det er vanlig å si at algoritmen er effektivt hvis kjøretiden er $f(k) * p(n)$ der f er en vilkårlig begrenset funksjon og p er et polynom. Legg merke til at kjøretiden er eksponentiell kun i k og ikke i inndata-størrelsen n direkte. Slike problem kaller vi FPT (fixed parameter tractable) for gitt parameter. Dette er en litt problematisk definisjon siden kjøretiden avhenger så sterkt av k mens det ikke er noen begrensinger på k. For en lav k er dette en effektiv algoritme, spørsmålet hvor mange ikke trivielle inndata som har lav verdi for k. Hvis $f(k) >> p(n)$ for nesten alle ikke trivielle inndata, fører det til

en eksponentiell algoritme, men hvis vi klarer å finne en parameter som er liten for mange inndata og som samtidig fører til en FPT algoritme, får vi polynomisk algoritme for alle disse inndataene og eksponentiell for resten av inndata.

En annen måte å se parametrisering på er å dele alle mulige inndata inn i grupper, en gruppe for hver verdi av parameteren k, der $k \in 0, 1, 2, ..., n$. Det er da meningen at k sier hvor vanskelig det er å løse problemet for den gruppen av inndata. La oss for eksempel prøve med problemet å finne en hamiltonsk sykel. Gruppe k kan være alle grafer som har maks $k * (n-1)/2$ kanter. Siden maks kanter er $n*(n-1)/2$ kanter er største verdi parameteren n. Vil dette hjelpe oss?

1. $k = 0$ har ingen kanter og derfor ingen sykel.
2. $k = 1$ har $(n-1)/2$ kanter og er derfor ikke sammenhengende.
3. $k = 2$ har $n-1$ kanter, en hamiltonsk sykel har n.
4. $k = 3$ for disse grafene er problemet NP-komplett.

Allerede for $k = 3$ får vi problemer med å løse problemet. Dette vises ved reduksjon fra hamiltonsk sti ved å legge til en sti av lengde n^2 mellom de to nodene som er start og slutt. Det betyr at vi valgte en parameter som ikke gir oss en god inndeling. Denne parameteren gir ikke en FPT algoritme, men kanskje vi kan finne en bedre parameter som gir en FPT algoritme. I neste avsnitt beskrives en parameter som for mange problemer gir oss FPT algoritmer.

5.1 Trebredde

Trebredde er en parameter som for veldig mange problem fører til en FPT algoritme, intuitivt er det en parameter mellom 0 og n som sier hvor mye en graf på n noder ligner på et tre. Slik at alle grafer som inneholder minst en sykel får trebredde minst 2. Jeg har valgt å gi en rekursiv definisjon av trebredde fordi den illustrerer poenget med dette avsnittet best, det er mange andre måter å definere trebredde som er vanligere.

Reduksjon av en node v betyr å gjøre alle naboene til v om til en klikk og så fjerne v.

Definisjon 5.1. *(trebredde)*
En graf på 1 *node har trebredde* 0*. En graf* $G = (V, E)$ *har trebredde* $\leq k$ *hvis og bare hvis det finnes en node* $v \in V$ *med gradtall* $\leq k$ *slik at grafen* H*, som er et resultat av å redusere grafen* G *med node* v*, har trebredde* $\leq k$*.*

Å bestemme trebredden er et spørsmål om å velge den riktige rekkefølgen av noder slik at reduksjon av nodene i denne rekkefølgen aldri prøver å redusere en node med gradtall $> k$ og fører til at det bare er en node igjen. Den minste k som gjør dette mulig er trebredden til grafen. Ingen node kan ha gradtall $> n - 1$ derfor er trebredden alltid $< n$.

Teorem 5.1. *[2]*
Å avgjøre om en graf har trebredde $\leq k$ *kan gjøres i tid* $O(f(k) * n)$*.*

Dette er viktig for første del av en FPT algoritme, nemlig å sjekke om inndatagrafen har trebredde $\leq k$. Andre del er selvfølgelig avhengig av problemet. Trebredde har de siste 10-20 årene ført til utallige FPT algoritmer og er et av de viktigste resultatene innen parametrisert kompleksitet. En av de viktige årsakene til trebredde sin sterke posisjon er følgende teorem:

Teorem 5.2. *[1]*
Ethvert grafproblem som kan uttrykkes i "Monadic second order logic" (MSOL) kan løses i FPT tid parametrisert ved trebredden til grafen.

Hvis vi kunne vise at de fleste naturlige nettverk har begrenset trebredde ville det være stor sannsynlighet for at mange av de problemene vi ønsker å løse på naturlige nettverk kan løses effektivt. Men det er likevel ikke sikkert vi kan bruke slike algoritmer på store nettverk som webgrafen siden $f(k)$, selv om det er en konstant, kan være stor og selv lineære algoritmer tar lang tid på webgrafen. Men på noen nettverk kan FPT algoritmer være svært nyttige.

Trebredde er ikke anvendelig for mange problemer innen naturlige nettverk siden grafen som beskriver nettverket er rettet. Vi kan se bort fra retningen på kanten og beregne trebredden, men da mistes viktig informasjon om det naturlige nettverket. Kelly-bredde er et forslag til en parameter for rettede grafer som skal være en analog til trebredde for rettede grafer.

5.2 Kelly-bredde

Det har kommet mange forslag for en rettet analog til trebredde.

- Directed treewidth
  Johnson, Robertson, Seymour and Thomas (2001)

- D-width
  Safari (2005)

- DAG-width
  Berwanger, Dawar, Hunter and Kreutzer (2006), Obdržálek (2006)

- Kelly-width
  Hunter and Kreutzer (2007)

Vi mener Kelly-bredde er det beste forslaget så langt. For å begrunne dette ser vi på likheter mellom trebredde og Kelly-bredde. En av likhetene mellom trebredde og Kelly-bredde vises av følgende definisjon som finnes i [12] og også i vår artikkel i Appendiks B:

Definisjon 5.2. *(Kelly-bredde)*
En graf på 1 *node har Kelly-bredde* 1*. En graf* $G = (V, E)$ *har Kelly-bredde* $\leq k$ *hvis og bare hvis det finnes en node* $v \in V$ *med utgradtall* $\leq k - 1$ *slik at grafen* H*, som er et resultat av å redusere grafen* G *med node* v*, har Kelly-bredde* $\leq k$*.*

Denne definisjonen er ganske lik trebredde, selv om reduksjon av en node v i en rettet graf er litt annerledes enn i en urettet graf. For alle innaboer av v, legg til en kant til hver utnabo av v, men ingen kanter kan ha samme start og sluttnode (ingen løkker). Derfor er det håp om at dette vil gi flere nye FPT algoritmer for problemer med rettede grafer som inndata.

For at Kelly-bredde skal være nyttig må vi effektivt kunne avgjøre om en graf har Kelly-bredde k, det er ennå uvisst hvor vanskelig dette er. Vedlagt i Appendiks er en artikkel som viser hvordan grafer med Kelly-bredde ≤ 2 kan gjenkjennes. Det er vanskelig å si om Kelly-bredde er en god parameter for naturlige nettverk, særlig siden vi ikke en gang vet presist hva naturlige nettverk er.

6 Oppsummering og videre arbeid

Vi har ikke fått et endelig svar på hva naturlige nettverk er, men kanskje fått en litt bedre forståelse av begrepet. Det vil nok ta tid før vi oppnår enighet om dette spørsmålet. Vi trenger mer forståelse av naturlige nettverk for å få enighet. Det å prøve forskjellige egenskaper og se hvor vanlig det er at naturlige nettverk har disse egenskapene er en måte å angripe problemet på. Siden det er så lett å samle informasjon om naturlige nettverk via Internett, det kan være webgrafen selv eller via de nye populære sidene for sosiale nettverk som "Facebook" og "My space" så gir dette informatikere en gylden mulighet til å utføre spennende forskning.

Et interessant fremtidig forskningsprosjekt er å analysere Kelly-bredden til forskjellige naturlige nettverk, for å se om ideene i kapittel 5 har noe for seg. Det forutsetter da at vi klarer å beregne Kelly-bredden effektivt.

Referanser

[1] Stefan Arnborg, Bruno Courcelle, Andrzej Proskurowski, and Detlef Seese. An algebraic theory of graph reduction. *J. ACM*, 40(5):1134–1164, 1993.

[2] Hans L. Bodlaender. A linear time algorithm for finding tree-decompositions of small treewidth. *SIAM Journal on Computing*, 25:1305–1317, 1996.

[3] Béla Bollobás and Wenceslas Fernandez de la Vega. The diameter of random regular graphs. *Combinatorica*, 2(2):125–134, 1982.

[4] Andrei Broder, Ravi Kumar, Farzin Maghoul, Prabhakar Raghavan, Sridhar Rajagopalan, Raymie Stata, Andrew Tomkins, and Janet Wiener. Graph structure in the web. *Comput. Netw.*, 33(1-6):309–320, 2000.

[5] D. Cartwright and F. Harary. Structural balance: a generalization of Heider's theory. *Psychological Review*, 63(5):277–93, 1956.

[6] David Easley and Jon Kleinberg. Networks. Webpage, 2007. http://www.infosci.cornell.edu/courses/info204/2007sp/.

[7] J. Guare. *Six Degrees of Separation.* Vintage books, 1990.

[8] Jiong Guo, Jens Gramm, Falk Hüffner, Rolf Niedermeier, and Sebastian Wernicke. Compression-based fixed-parameter algorithms for feedback vertex set and edge bipartization. *J. Comput. Syst. Sci.*, 72(8):1386–1396, 2006.

[9] Frank Harary. On the notion of balance of a signed grph. *Michigan Math. Journal*, (2):143–146, 1953.

[10] Fritz Heider. Attitudes and cognitive organization. *The Journal of Psychology, Smith College*, (21):107–112, 1946.

[11] Falk Hüffner. Algorithm engineering for optimal graph bipartization. In *Proceedings of the 4th International Workshop on Experimental and Efficient Algorithms (WEA '05)*, volume 3503 of *LNCS*, pages 240–252. Springer, 2005.

[12] Paul Hunter and Stephan Kreutzer. Digraph measures: Kelly decompositions, games, and orderings. In *SODA '07: Proceedings of the eighteenth*

annual ACM-SIAM symposium on Discrete algorithms, pages 637–644, Philadelphia, PA, USA, 2007. Society for Industrial and Applied Mathematics.

[13] J. Kleinberg. The small-world phenomenon: An algorithmic perspective. In *Proceedings of 32nd ACM Symposium on Theory of Computing*, 2000.

[14] J. Kleinberg. Complex networks and decentralized search algorithms. In *Proceedings of the International Congress of Mathematicians (ICM)*, 2006.

[15] J. Travers and S. Milgram. An experimental study of the small world problem. *Sociometry*, 1969.

[16] D. J. Watts and S. H. Strogatz. Collective dynamics of 'small-world' networks. *Nature*, 393(6684):440–442, June 1998.

Appendiks

A Definisjoner

Vi bruker grafer mange steder i denne oppgaven.

Definisjon A.1. *En rettet graf $G = (V, E)$ der V er en nodemengde og $E \subseteq V \times V$ er en mengde med ordnede par av noder kallt kanter.*

Med graf mener vi urettet graf, det betyr at $(u, v) \in E \Leftrightarrow (v, u) \in E$ alså er parene uordnet i motsetning til en rettet graf der parene er ordnet.

- En sti av lengde k i en graf $G = (V, E)$ er en ordnet delmengde av V, $v_0, v_1, ...v_k$ der ingen node forekommer mer enn en gang, slik at $\forall i : 1 \leq i < k$ er $(v_i, v_i + 1) \in E$.
- En sykel er en sti av lengde minst 3, der første og siste noden er den samme, men alle andre nodene forekommer høyst en gang.
- Et tre er en graf uten sykler.
- En bipartitt graf er en graf der alle syklene har et jevnt antall kanter.
- I en rettet graf har en node u har utgradtall k hvis antall noder v slik at (u, v) er en kant er k og inngradtall k hvis antall noder v slik at (v, u) er en kant er k. I en graf er inn og utgradtall alltid lik derfor kalles det bare gradtall.
- En graf er k-regulær hvis alle noder har gradtall k.
- En klikk er en mengde noder der alle nodene er naboer med hverandre.

I tilegg til grafteori brukes en del begreper om kompleksitet. For å måle kjøretid brukes notasjon som $O()$, $\Omega()$ og $\Theta()$.

- $f(n) \in O(g(n))$ hvis det finnes konstanter c og n_0 slik at for alle $n > n_0$ er $|f(n)| < c * |g(n)|$.
- $f(n) \in \Omega(g(n))$ hvis det finnes konstanter c og n_0 slik at for alle $n > n_0$ er $|f(n)| > c * |g(n)|$.
- $f(n) \in \Theta(g(n))$ hvis $f(n) \in O(g(n))$ og $f(n) \in \Omega(g(n))$.
- $f(n) \in o(g(n))$ hvis ikke $f(n) \in \Omega(g(n))$.

For å sammenligne forskjellige problemer deles problemene inn i kompleksitetsklasser, det finnes mange slike, her nevner vi kun de klassene vi bruker i oppgaven.

Definisjon A.2 (NP)**.** *NP er en klasse som inneholder alle JA/NEI problemer som, hvis svaret er JA, har et sertifikat av polynomisk størrelse som verifiserer svaret.*

Definisjon A.3 (NP-hardt)**.** *Et problem er NP-hardt hvis og bare hvis en polynomisk algoritme for problemet medfører at alle problemer i NP har en polynomisk algoritme.*

Definisjon A.4 (NP-komplett)**.** *Et problem er NP-komplett hvis og bare hvis det er i NP og er NP-hardt.*

Definisjon A.5 ($\sharp$ P)**.** *En klasse av problemer der svaret er antall løsninger. Og for hver løsning finnes det et sertifikat av polynomisk størrelse som verifiserer løsningen.*

Definisjon A.6 ($\sharp P$-komplett)**.** *Et problem er $\sharp P$-komplett hvis det er i $\sharp P$ og en polynomisk løsning av problemet vil føre til en polynomisk løsning av et hvert problem i $\sharp P$.*

Merk at siden det finnes et problem i $\sharp P$ som er NP-hardt vil en polynomisk løsning av et $\sharp P$-komplett problem føre til en polynomisk løsning på et hvert NP-komplett problem. Dette medfører at alle $\sharp P$-komplette problemer er NP-harde. Derfor er $\sharp P$-komplette problemer vanskeligere enn NP-komplette problemer.

Definisjon A.7 (FPT - Fixed-parameter tractable)**.** *FPT er en klasse av problemer kombinert med en parameter k som del av inndata. Et problem er FPT for bestemt parameter k hvis det kan løses på tid $O(f(k) * n^c)$.*

Det er lett å finne en parameter som løser et hvert løsbart problem, nemlig la $k = n$. Det er derfor viktig at den parameteren som velges gir en lav verdi for en stor mengde av mulige inndata. Noen problemer som er NP-harde kan løses effektivt for mange inndata, men hvis det hadde vært slik at alle inndata ga en så lav verdi at $f(k)$ er begrenset av et polynom av n så ville problemet hatt en polynomisk løsning og dermed ville P = NP, det er derfor høyst sannsynlig at noen inndata vil parameteren få en verdi som er av orden n.

Recognizing digraphs of Kelly-width 2

Daniel Meister and Jan Arne Telle and Martin Vatshelle
Department of Informatics, University of Bergen, 5020 Bergen, Norway

28th January 2008

Abstract

Kelly-width is a parameter of digraphs recently proposed by Hunter and Kreutzer as a directed analogue of treewidth. We give an alternative characterization of digraphs of bounded Kelly-width in support of this analogy, and the first polynomial-time algorithm recognizing digraphs of Kelly-width 2. For an input digraph $G = (V, A)$ the algorithm outputs a vertex ordering and a digraph $H = (V, B)$ with $A \subseteq B$ witnessing either that G has Kelly-width at most 2 or that G has Kelly-width at least 3, in time linear in H.

1 Introduction

The tractability of large classes of NP-complete problems when parameterized by the treewidth of the input graph counts among the strongest results in algorithmic graph theory. The algorithms behind this tractability have two stages: first an algorithm computing treewidth, then an algorithm solving the specific problem using the tree-structure discovered in the first stage. See for example [3] for a recent overview of these algorithms. For directed graphs (digraphs) there have been several proposals for a parameter analogous to treewidth: 'directed treewidth' of Johnson, Robertson, Seymour, Thomas [7], 'D-width' of Safari [10], 'DAG-width' of Berwanger, Dawar, Hunter, Kreutzer [2] and independently Obdržálek [9], and 'Kelly-width' of Hunter and Kreutzer [6]. Which of these proposed parameters is the better analogue of treewidth? In this paper we give evidence in support of the Kelly-width parameter.

The success of a model depends on a balance between the modeling power, which measures how general its domain of application is, and the analytical power, which measures how good it is as an analytical tool. The two are typically in conflict. This is also the case for the above proposals for tree-like parameters of digraphs. The better the modeling power, e.g. the larger the class of digraphs that have bounded parameter value, the worse the analytical power, e.g. the smaller the chance of successfully emulating both stages of the algorithmic results for treewidth. We do not go into details of the modeling and analytical powers of each of the proposed digraph parameters, but simply note that from a purely algorithmic point of view there is no clear winner. How then to choose the digraph parameter which is the most natural directed analogue of treewidth? Note that while some concepts of undirected graphs have unambiguous natural translations to directed graphs, e.g. from paths to directed paths, there are other concepts, e.g. cliques and separators, for which the translation is less clear. The treewidth parameter is known to have many equivalent characterizations. If we start with a characterization of treewidth that uses only concepts that have unambiguous translations to directed graphs then we should arrive at a directed graph parameter which is a natural analogue of treewidth. Hunter and Kreutzer did just this for an elimination

process of undirected graphs that defines treewidth to obtain a characterization of digraphs of bounded Kelly-width. We establish a second such correspondence in Section 3, a characterization of digraphs of Kelly-width at most k that is an analogue of a characterization of treewidth [11, 4].

We also enhance the algorithmic argument in favour of Kelly-width. Digraphs of Kelly-width 1 are the directed acyclic graphs and recognizable by a simple algorithm. For all larger values of k the only algorithms that were known for recognizing digraphs of Kelly-width k had running time exponential in the size of the input digraph [6]. We present a fast algorithm recognizing digraphs of Kelly-width 2 in Section 4. For an input digraph $G = (V, A)$ this algorithm will output a vertex ordering and a digraph $H = (V, B)$ with $A \subseteq B$ witnessing either that G has Kelly-width at most 2 or that G has Kelly-width at least 3, in time linear in H. In the positive case the witness can be used to easily find a decomposition of the digraph into a tree-like structure. Section 5 discusses Kelly-width k recognition for $k > 2$ before some final remarks in Section 6.

2 Graph preliminaries and digraphs of bounded Kelly-width

We consider simple finite directed graphs, "digraphs" for short. A digraph is denoted as $G = (V, A)$ where V is the vertex set and A the arc set of G. In particular, G does not have loops. When we deal with undirected graphs, we will explicitly mention it. An arc of graph G is denoted as (u, v) with u an in-neighbor of v and v an out-neighbor of u. Further definitions are given when they are needed.

Hunter and Kreutzer introduced the notion of Kelly-width [6]. Kelly-width is a parameter for digraphs, and it is the least width of a so-called *Kelly-decomposition*. We will not define Kelly-decompositions here, since we will not use this notion. The authors gave several characterizations of digraphs of bounded Kelly-width by: elimination process, inductive construction, cops-and-robber game. We will study digraphs of bounded Kelly-width starting from the elimination process.

Undirected graphs of bounded treewidth have a nice characterization using an elimination process. Let x be a vertex of an undirected graph G. The operation *reducing G by x* yields graph G' that is obtained from G by deleting vertex x and adding an edge between each pair of non-adjacent neighbors of x, in other words, making its neighborhood into a clique.

Theorem 2.1 (folklore) *An undirected graph G has treewidth at most k if and only if G can be reduced to the empty graph by repeatedly reducing by a vertex of degree at most k.*

The operation of reducing by a vertex was translated by Hunter and Kreutzer into the world of digraphs as follows: *reducing digraph G by x* yields digraph G' that is obtained from G by deleting vertex x and adding arcs from all in-neighbors of x to all out-neighbors of x, but no loops or multiple edges. The following result will for our purposes serve as the definition of Kelly-width.

Theorem 2.2 ([6]) *A digraph G has Kelly-width at most $k + 1$ if and only if G can be reduced to the empty digraph by repeatedly reducing by a vertex of out-degree at most k.*

The elimination process of Theorem 2.2 defines a vertex ordering, where the first vertex is the vertex eliminated first. We call this an *elimination ordering* of width at most k. Note that it is implicit in Theorem 2.2 that for every digraph G and every subgraph H of G, the Kelly-width of H is at most the Kelly-width of G.

3 An analogy between Kelly-width and treewidth

In this section, we show that digraphs of bounded Kelly-width are the digraphs whose vertices can be arranged in a linear order to satisfy special conditions, without using the notion of vertex reductions. We start with a lemma that is implicit, but not proven, in [6]. When we speak of paths, we mean paths that visit a vertex at most once.

Lemma 3.1 *Let $G = (V, A)$ be a digraph, and let $X \subseteq V$. Reduce G by the vertices in X, taken in any order, and obtain H. Then, (v, w) is arc in H if and only if there exists a v, w-path in G using only vertices from $X \cup \{v, w\}$.*

Proof We show the lemma by induction over the cardinality of X. If $|X| = 0$ then the claim obviously holds, since G has a v, w-path using only vertices from $\emptyset \cup \{v, w\}$ if and only if $(v, w) \in A$. Now, let the claim be true for every subset of V of cardinality at most $k \geq 0$. Let $X \subseteq V$ where $|X| = k + 1$. Let $x \in X$. Obtain G' by reducing G by the vertices in $X \setminus \{x\}$. Obtain H by reducing G' by x. Let v, w be two vertices of H.

($\Rightarrow$) Let $(v, w) \in A(H)$. If $(v, w) \in A(G')$ then there is a v, w-path in G using only vertices from $X \cup \{v, w\}$ by induction hypothesis. Otherwise, by the definition of the reduction operation, $(v, x), (x, w) \in A(G')$. Due to induction hypothesis, there are v, x- and x, w-paths in G using only vertices from $(X \setminus \{x\}) \cup \{v, x\}$ and $(X \setminus \{x\}) \cup \{x, w\}$, respectively. Hence, there is a v, w-path in G using only vertices from $(X \setminus \{x\}) \cup \{v, x\} \cup \{x, w\} = X \cup \{v, w\}$ in G.

($\Leftarrow$) Let there be a v, w-path P in G using only vertices from $X \cup \{v, w\}$. If P does not contain x then P contains only vertices from $(X \setminus \{x\}) \cup \{v, w\}$ and $(v, w) \in A(G') \subseteq A(H)$ due to induction hypothesis. Otherwise, if x is a vertex on P, then there are v, x- and x, w-paths in G using only vertices from $(X \setminus \{x\}) \cup \{v, x\}$ and $(X \setminus \{x\}) \cup \{x, w\}$, respectively. By induction hypothesis, $(v, x), (x, w) \in A(G')$, and hence $(v, w) \in A(H)$.

Thus, the lemma follows. □

Note that Lemma 3.1 yields as a result that the graph obtained from reducing by a set of vertices does not depend on the actual order in which the reduction is executed.

The following characterization of undirected graphs of bounded treewidth was first given explicitly by Thorup [11] (and is implicit in the work of Dendris et al. [4]):

Lemma 3.2 ([11, 4]) *An undirected graph G has treewidth at most k if and only if it has a vertex ordering $\sigma = v_1, v_2, \ldots, v_n$ and, for each $1 \leq i \leq n$, a set $S_i \subseteq \{v_1, v_2, \ldots, v_{i-1}\}$ of at most k vertices such that there is no path in $G \setminus S_i$ from v_i to a vertex in $\{v_1, v_2, \ldots, v_{i-1}\}$.*

We prove an analogous result for digraphs and Kelly-width:

Lemma 3.3 *A digraph G has Kelly-width at most $k+1$ if and only if it has a vertex ordering $\sigma = v_1, v_2, \ldots, v_n$ and, for each $1 \leq i \leq n$, a set $S_i \subseteq \{v_1, v_2, \ldots, v_{i-1}\}$ of at most k vertices such that there is no path in $G \setminus S_i$ from v_i to a vertex in $\{v_1, v_2, \ldots, v_{i-1}\}$.*

Proof ($\Rightarrow$) Let G have Kelly-width at most $k + 1$. According to Theorem 2.2, there is an elimination ordering of width at most k for G; let $\sigma = v_1, \ldots, v_n$ be the reversal of such an

ordering. Let $i \in \{1, \ldots, n\}$. Let P be a v_i, w-path for $w \in \{v_1, \ldots, v_{i-1}\}$. Let u be the first vertex from $\{v_1, \ldots, v_{i-1}\}$ on P. Let H be obtained from G by reducing by the vertices in $\{v_{i+1}, \ldots, v_n\}$. Due to Lemma 3.1, (v_i, u) is an arc in H. Let S_i be the set of out-neighbors of v_i in H. Due to Theorem 2.2 and Lemma 3.1, $|S_i| \leq k$. Then, there is no v_i, w-path for $w \in \{v_1, \ldots, v_{i-1}\}$ in $G \setminus S_i$.

($\Leftarrow$) Let $\sigma = v_1, \ldots, v_n$ be a vertex ordering for G and $S_1, \ldots, S_n$ the corresponding sets of vertices satisfying the claim. We show that $v_n, \ldots, v_1$ is an elimination ordering for G of width at most k. Let $i \in \{1, \ldots, n\}$ and let H_i be the result of reducing G by $\{v_{i+1}, \ldots, v_n\}$. For every out-neighbor x of v_i in H, there is a v_i, x-path in G using only vertices from $\{x, v_i, \ldots, v_n\}$ according to Lemma 3.1. Thus, $x \in S_i$. Hence, v_i has at most k out-neighbors in H_i, and the claim follows. □

4 Algorithm for recognition of digraphs of Kelly-width 2

Theorem 2.2 gives an easy non-deterministic algorithm for recognition of digraphs of Kelly-width $k+1$: reduce by some vertex of out-degree at most k and halt with a positive answer if the result is an empty graph. A polynomial-time algorithm does not evolve from this, since it is not clear which of the possible vertices to choose. However, we show that for Kelly-width 2 any vertex of out-degree at most 1 can safely be chosen.

Algorithm 1 Kelly-width 2 recognition

Input: Digraph G
Output: 'Yes', if G has Kelly-width at most 2, and 'No' otherwise

while G has a vertex v of out-degree at most 1 **do**
  reduce by v
end while;
if G is the empty digraph **then** return 'Yes' **else** return 'No' **end if**

Theorem 4.1 *The Kelly-width 2 recognition algorithm is correct.*

Proof By Theorem 2.2 it is clear that the algorithm will never answer Yes unless G has Kelly-width at most 2. We can thus assume that G has an elimination ordering establishing that it has Kelly-width at most 2. We show that any vertex v of out-degree at most 1 can be the first in such an elimination ordering. The correctness of the algorithm will follow by induction.

Assume v of out-degree at most 1 and let σ be an elimination ordering establishing Kelly-width at most 2 in which v is reduced the earliest possible. If v is first we are done, so assume for a contradiction that there is a vertex u reduced immediately before v. The vertex u has out-degree at most 1 when it is about to be reduced. By assumption the vertex v has out-degree at most 1 in G and since reducing vertices of out-degree at most 1 cannot increase the out-degree of any vertex, v has out-degree at most 1 also when u is about to be reduced. Let σ' be the elimination order we get from σ by simply swapping the order of u and v. Thus, reducing G in the order given by σ' will up to reduction of u obey the rule of always reducing by vertices of out-degree at most 1. By Lemma 3.1 the reduced graphs after reducing either by σ up to v or by σ' up to u are equal, and thus reducing G by σ' is also an elimination ordering establishing Kelly-width at most 2. This contradicts the choice of σ as v is reduced later in σ than in σ'. □

The algorithm can be provided with an output certificate guaranteeing correctness of an implementation, as follows. Let $\sigma = v_1, v_2, \ldots, v_i$ be the reduction sequence found by the algorithm, with $i = n$ if and only if the input graph had Kelly-width at most 2. Let F be the set of edges added during vertex reduction. If the input graph has Kelly-width at most 2 then certainly by Theorem 2.2 the order σ and the edges F together with the input graph form a certificate of this. Also in the negative case this is a certificate, as the proof of Theorem 2.2 implies that reducing by vertices of out-degree at most 1 in any order is safe, and σ does just this adding edges F to result in a reduced graph with no vertices of out-degree at most 2.

Theorem 4.2 *The Kelly-width 2 recognition algorithm augmented with output σ and $H = G \cup F$ is a certifying algorithm that, given a digraph G, decides whether G has Kelly-width at most 2. The running time and the working space of the algorithm are linear in the size of the output graph.*

Proof We have already argued that the algorithm is correct and that the output can be used as a certificate. For the running time of the reduction algorithm note that, after every reduction step, the resulting graph is a subgraph of H. Reducing by a vertex takes time linear in the number of is neighbors, since the in-neighbors of the reduced vertex become in-neighbors of the single out-neighbor, if there is one. Let a be eliminated and let b be its only out-neighbor. The crucial point is to find the in-neighbors of a that are in-neighbors also of b. These are exactly the vertices whose out-degree is decreased by 1. We assume that the adjacency lists of the vertices are ordered with respect to some ordering. The intersection of the two in-neighborhoods is computed by just scanning the two lists. The reason that this is linear time even if the in-neighborhood list of b is larger than that of a is that the in-degree of a in H is not smaller than the in-degree of b. This gives linear running time and working space in the size of input and output graph. □

For deciding whether a graph has Kelly-width exactly 2, it suffices to run the algorithm only for non-acyclic digraphs. Acyclic digraphs are exactly the digraphs of Kelly-width 1. The algorithm has running time linear in the output certificate. We leave it as an open problem if the algorithm can be implemented to have running time linear in the input digraph. We see at least two reasons why this will not be trivial.

Firstly, consider the digraph G constructed from a bi-directional path $v_1, v_2, \ldots, v_n$ by adding arcs from every vertex into v_1. It is easy to check that $\sigma = v_1, v_2, \ldots, v_n$ is the one and only elimination ordering showing that G has Kelly-width 2. When reducing by vertex v_i in this order, we will add $n - i - 2$ new arcs. Thus, any elimination process showing that G has Kelly-width 2 adds $\Theta(n^2)$ arcs even though the input graph has only $\Theta(n)$ arcs.

Secondly, not all digraphs of Kelly-width 2 have an elimination ordering $\sigma = v_1, v_2, \ldots, v_n$ with no added arcs going 'forward' from a lower-numbered to a higher-numbered vertex. Consider the digraph G depicted in Figure 1. Elimination order d, b, f, a, c, e shows that G has Kelly-width 2, but note that a forward arc (f, a) is then added. Let us argue that every elimination order showing Kelly-width 2 will have such forward arcs added. G has three vertices with out-degree at most 1, namely a, b and d. Furthermore, vertices c and e have out-degree 5, which means that c and e are the last two vertices in every such order. We distinguish two cases. If a is the first vertex then the arc (f, e) is added which must be a forward arc as we have already argued that e must be reduced after f. If b or d is the first vertex then the arc (f, a) is added so a would need to be reduced before f but then after reducing by a (and the other of b or d) we would have a reduced graph with every vertex having at least two out-neighbors.

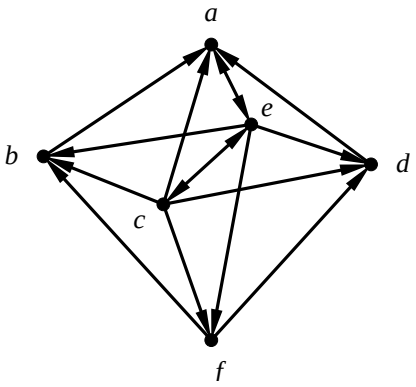


Figure 1: Elimination order d, b, f, a, c, e shows that this digraph has Kelly-width 2

5 Recognition of digraphs of Kelly-width k

As for recognition of graphs of Kelly-width k for any $k > 2$, we do not believe these ideas will generalize easily. The main reason for this is that when reducing by a vertex of out-degree larger than 1 the out-degree of remaining vertices can increase. Nevertheless, it is easy to see that for any k we can safely reduce by a vertex of in- and out-degree 1. As another simplification, we show in the following that it suffices to consider only strong-connected components for determining the Kelly-width. For a digraph $G = (V, A)$, a *directed cut* (L, R) is a partition of the vertex set into non-empty sets L and R such that there is no arc (u, v) in G with $u \in R$ and $v \in L$.

Lemma 5.1 *Let $G = (V, A)$ be a digraph and let $k \geq 0$. Let (L, R) be a directed cut in G, and let G_L and G_R be the subgraphs of G induced by L and R, respectively. Then, the Kelly-width of G is at most $k + 1$ if and only if the Kelly-width of G_L and G_R is at most $k + 1$.*

Proof Since the Kelly-width does not increase by taking subgraphs, Kelly-width of G at most $k+1$ implies that the Kelly-width of G_L and G_R is at most $k+1$. For the other direction, let the Kelly-width of G_L and G_R be at most $k + 1$. Due to Theorem 2.2, there are elimination orderings σ_L and σ_R of width at most k for G_L and G_R, respectively. We show that the concatenation of both orderings, $\sigma_R \circ \sigma_L$, is an elimination ordering of width at most k for G. Since there is no path from vertices in R to vertices in L in G, all arcs that are added to G when reducing according to σ_R are arcs that are added to G_R when reducing according to σ_R. Thus, when reducing G according to σ_R, no vertex has out-degree more than k when it is chosen. Furthermore, due to Lemma 3.1, the result of reducing G by R is equal to G_L. Since σ_L is an elimination ordering for G_L of width at most k, we conclude that $\sigma_R \circ \sigma_L$ is an elimination ordering for G of width at most k, and thus G has Kelly-width at most $k + 1$. □

Corollary 5.2 *Let $G = (V, A)$ be a digraph. Then, the Kelly-width of G is equal to the maximum Kelly-width taken over the strongly-connected components of G.*

Proof If G is strongly-connected the claim obviously holds. Now, let G not be strongly-connected. Then, there is a strongly-connected component H of G such that no vertex of H has an out-neighbor outside H. Hence, $(V(G) \setminus V(H), V(H))$ is a directed cut in G. We apply Lemma 5.1 and obtain the result by induction. □

6 Final remarks

We gave an easy algorithm for recognition of digraphs of Kelly-width 2. Our algorithm for Kelly-width 2 can be considered a directed version of the undirected counterpart: graphs of treewidth 2 are the graphs reducible to the empty graph by iteratively reducing vertices of degree at most 2. Also graphs of treewidth 3 have a recognition algorithm based on vertex reduction, albeit with more complicated rules for choosing which vertex to reduce, as discovered by Arnborg et al. [1]. It is an interesting open problem if also recognition of graphs of Kelly-width 3 can be done by reduction rules. A more general open problem is to decide if Kelly-width is FPT.

Acknowledgements. We thank an anonymous referee for suggesting simplified proofs.

References

[1] S. Arnborg, A. Proskurowski, and D. Corneil. Forbidden minors characterization of partial 3-trees. *Discrete Mathematics* 80(1): 1-19 (1990)

[2] D. Berwanger, A. Dawar, P. Hunter, and St. Kreutzer. DAG-Width and Parity Games. In *Proceedings of the 23rd Annual Symposium on Theoretical Aspects of Computer Science, STACS 2006*, volume 3884 of *Lecture Notes in Computer Science*, pages 524–536. Springer-Verlag, 2006.

[3] H.L. Bodlaender. Treewidth: Characterizations, Applications, and Computations. In *Proceedings of the 32nd International Workshop on Graph-Theoretic Concepts in Computer Science, WG 2006*, volume 4271 of *Lecture Notes in Computer Science*, pages 1–14. Springer-Verlag, 2006.

[4] N.D. Dendris, L.M. Kirousis, and D.M. Thilikos. Fugitive-search games on graphs and related parameters. *Theoretical Computer Science*, 172:233–254, 1997.

[5] G.A. Dirac. On rigid circuit graphs. *Abhandlungen aus dem Mathematischen Seminar der Universität Hamburg*, 25:71–76, 1962.

[6] P. Hunter and St. Kreutzer. Digraph Measures: Kelly Decompositions, Games, and Orderings. In *Proceedings of the Eighteenth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2007*, pages 637–644, 2007.

[7] Th. Johnson, N. Robertson, P.D. Seymour, and R. Thomas. Directed Tree-Width. *Journal of Combinatorial Theory, Series B*, 82:138–154, 2001.

[8] D. Meister, J.A. Telle, and M. Vatshelle. Characterization and recognition of digraphs of bounded kelly-width. Technical Report 351, Institutt for Informatikk, Universitetet i Bergen, 2007.

[9] J. Obdržálek. DAG-width – Connectivity Measure for Directed Graphs. In *Proceedings of the Seventeenth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2006*, pages 814–821, 2006.

[10] M.A. Safari. D-Width: A More Natural Measure for Directed Tree Width. In *Proceedings of the 30th International Symposium on Mathematical Foundations of Computer Science, MFCS*

2005, volume 3618 of *Lecture Notes in Computer Science*, pages 745–756. Springer-Verlag, 2005.

[11] M.Thorup. All Structured Programs have Small Tree-Width and Good Register Allocation. In *Information and Computation*, 142(2): 159-181 (1998)