Netværkskodning

Speciale af Louise Foshammer og Malte Neve-Græsbøll

AALBORG UNIVERSITET

AALBORG UNIVERSITET

Institut for Matematiske Fag
Fredrik Bajers Vej 7G
Telefon 99 40 99 40
www.math.aau.dk

Titel:
Netværkskodning

Projektperiode:
Forårssemesteret 2014

Projektgruppe:
G2-104a

Deltagere:

Louise Foshammer

Malte Neve-Græsbøll

Vejleder:
Olav Geil

Oplagstal: 7
Sidetal: 138
Afsluttet den 10. juni 2014

Synopsis:

I dette projekt vil vi betragte netværkskodningsproblemer. Vi tager udgangspunkt i lineær multicast netværkskodning og introducerer grundlæggende teori, betragter hvornår et sådant netværkskodningsproblem er løseligt og sandsynligheden for at finde en løsning ved tilfældigt valg af indkodningskoefficienterne. Derefter vil vi undersøge hvordan man kan beskytter sig imod fejl på beskeder eller at beskeder forsvinder helt. Dette vil vi gøre med Kötter-Kschischang koder, som har den egenskab, at de kan rette fejl introduceret i netværket og dekode selv hvis enkelte beskedpakker forsvinder i netværket. Vi vil herefter arbejde med lineære koder i generelle netværk og vise en metode, der ved hjælp af Gröbnerbaser undersøger om der eksisterer en lineær løsning til netværkskodningsproblemet. Vi vil slutteligt undersøge tilfælde hvor lineær netværkskodning ikke er tilstrækkeligt, og prøve at finde løsningsstrategier for sådanne netværk. Først vil vi forsøge at sætte nogle begrænsninger og krav for de enkelte indkodningspunkter, derefter vil vi undersøge muligheden for tilfældigt at vælge funktioner, som løser netværkskodningsproblemet, i forskellige funktionsklasser.

Abstract

Network coding is a field in rapid development. This thesis focuses in on different, but related, topics within this field.
First and foremost we seek to explain the basic theory about networks and network coding. We focus our attention on linear network coding for multicast and tackle this by using random network coding. This means that we randomly decide what to do in each step of the network and then figure out if these decisions together solves the problem. The probability of choosing the combinations correctly in every step of the network is close to one for a field large enough. We present an algorithm for choosing the combinations randomly. The algorithm will change the chosen strategy in a vertice if it happens to be chosen such that the network can't be solved. Furthermore we create matrices according to the topology of the network. From these matrices we create transfermatrices from which we derive a transferpolynomial, which can tell us if a network coding problem is solvable.
After this we move on to subspace codes. We are still looking at linear network coding for multicast, but now we introduce the possibility of errors and erasures happening in the network. It is possible to create such codes, and they are rather resistant towards errors in the sense that an error cannot propagate through the network and spread in this way and no matter how many errors happen on one vector it only counts as one. Still it seems that because an error can lead to an erasure it diminishes the usability of the codes for some networks. It might be possible to remedy this somehow, but as for now the codes only work optimally in certain networks.
We then turn our attention to linear network coding in general and expands the basic theory to fit the case when multicast is not assumed. We can create the same matrices as before even though we are not able to create just one transferpolynomial to see if the network coding problem is solvable. We are however able to set up a system of equations which must be fullfilled if the

network coding problem is to be solvable. This further leads us to show a way to decide whether or not a network coding problem is solvable using the Gröbner basis of the network.
Lastly we turn our attention to general network coding. We try to solve the network by choosing randomly what functions to use in the encoding vertices. The probability of choosing correctly appears to be small using every imaginable function, which is why we attempt to diminish the number of possible functions to choose between. We try to do this in two different ways. First of all we try to limit the type of functions that we can choose between and secondly we try to limit the functions by analyzing the network and deciding on certain properties that are needed in the different vertices.

Forord

Dette speciale er skrevet indenfor retningen Diskret Matematik ved Institut for Matematiske Fag, Aalborg Universitet. Specialet omhandler emnet netværkskodning og er delvist en overbygning på det projekt, som forfatterne skrev på 9.semester og som også behandlede emnet netværkskodning. Projektet kan findes i universitetets projektbibliotek [5]. Kapitel 1 er en meget redigeret udgave af et kapitel fra det tidligere projekt, mens kapitlerne 5 og 6 består af en gennemrettet udgave af noget tekst fra det tidligere projekt.
Målet med specialet er at belyse forskellige emner indenfor netværkskodning. Specialet spænder bredt fra tilfældig netværkskodning i lineære multicast netværkskodningsproblemer over fejl- og sletningskorrigerende koder i det lineære multicast netværkskodningsproblem videre til at finde ud af om lineære netværkskodningsproblemer kan løses over til generelle netværkskodningsproblemer og muligheden for at behandle disse.
Specialet forudsætter et grundlæggende kendskab til grafteori samt lineær og abstrakt algebra.
Som en del af at skrive dette speciale har vi samarbejdet med "Network Coding Focus Group" (NCFG), som er en fokusgruppe ved Aalborg Universitet. Vi leverede det teoretiske grundlag for Kötter-Kschischang koder, som er en form for fejl- og sletningskorrigerende koder. NCFG ville forsøge at udnytte de fejl- og sletningskorrigerende egenskaber ved koderne til at optimere i forbindelse med distributed storage.

Læsevejledning

Specialet er inddelt i kapitler, svarende til forskellige områder indenfor netværkskodning.
I kapitel 1 introducerer vi grundlæggende koncepter indenfor netværkskodning,

vi introducerer netværkskodningsproblemet med lineær netværkskodning og multicast netværkskodningsproblemer som vores primære fokus.
Kapitel 2 introducerer emner indenfor abstrakt algebra, der skal benyttes som teoretisk grundlag for de følgende kapitler.
I kapitel 3 betragter vi Kötter-Kschischang koder, som er lineære underrumskoder for multicast netværkskodningsproblemer. Vi fortsætter altså som sådan emnet netværkskodning, hvor vi slap det ved de lineære multicast netværkskodningsproblemer i kapitel 1.
I kapitel 4 derimod bevæger vi os væk fra multicast netværkskodningsproblemer, selvom vi stadig fastholder, at det skal være lineære netværkskoder. Vi betragter i dette kapitel måder hvorpå vi kan udvide noget af teorien fra kapitel 1 til også at gælde for generelle netværk, og forsøger ved hjælp af teorien fra kapitel 2 at opstille teori for hvornår et netværkskodningsproblem kan løses.
Kapitel 5 tager os endelig til det helt generelle netværkskodningsproblem. Vi forsøger her at imitere noget af det tidligere teori og forsøger at vælge en kode tilfældigt ud fra de mulige. Vi forsøger herefter på to forskellige måder, at indskrænke mængden af mulige koder vi vælger imellem.

Indhold

Kapitel 1

Introduktion

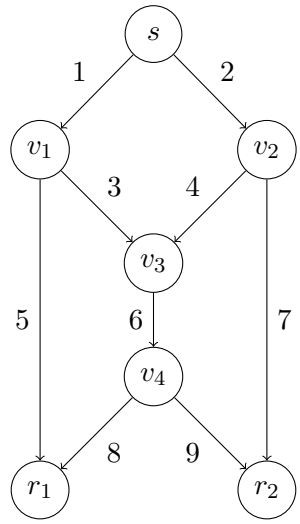
Dette kapitel er baseret på en artikel af Geil og Thomsen [7] og på en artikel af Kötter og Médard [13].
I dette kapitel ønsker vi at introducere nogle grundbegreber indenfor netværkskodning. Lad os begynde med at betragte et grundlæggende eksempel. På figur 1.1 ses den orienterede graf $G = (V, E)$, hvor kanterne er nummereret $1, \ldots, |E| = 9$. Afsenderen s ønsker at sende en mængde $\{a, b\}$ af beskeder, hvor a og b er tegn fra alfabetet $\mathbb{F}_2$, til modtagerne r_1 og r_2.
Vi betragter først modtager r_1 og finder to kantdisjunkte veje fra s til r_1

$$\{(1, 5), (2, 4, 6, 8)\}. \tag{1.1}$$

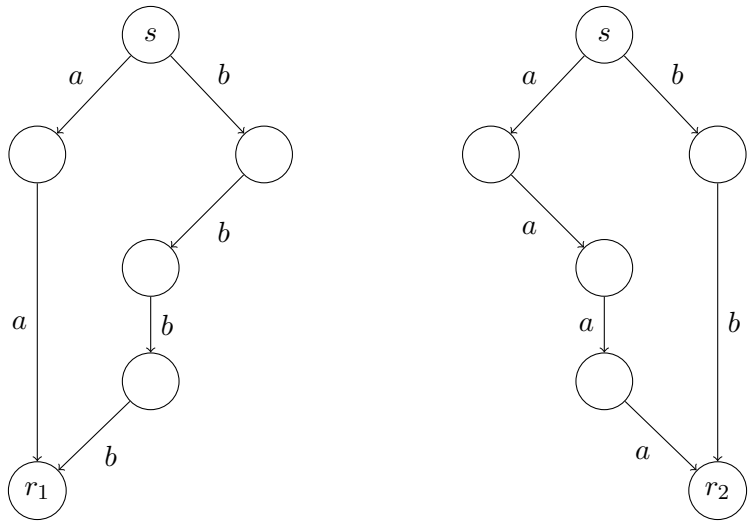
Mængden, der indeholder veje fra afsender til modtager, kaldes et flow og i dette tilfælde har det størrelse to. Et flow kan bruges til at få information fra afsender til modtager. I eksemplet sender s meddelelse a på kant 1 og meddelelse b på kant 2. Meddelelse a følger så vejen $(1, 5)$ til r_1 og meddelelse b følger vejen $(2, 4, 6, 8)$.
For r_2 er flowet

$$\{(1, 3, 6, 9), (2, 7)\}. \tag{1.2}$$

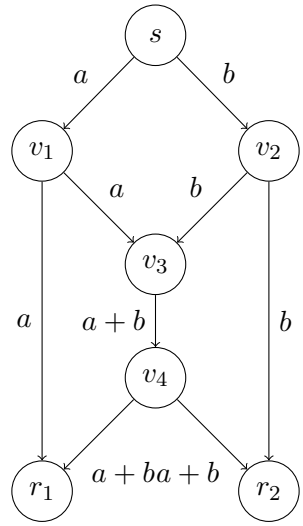
Dette er illustreret på figur 1.2. Nu har vi altså to partielle løsninger til det fulde netværkskodningsproblem, at få meddelelse a og b fra s frem til både r_1 og r_2 samtidig. Hvis vi prøver at kombinere de to løsninger ser vi, at kant 6 ifølge flow (1.1) skal sende meddelelse b og ifølge (1.2) meddelelse a.
De to partielle løsninger kan altså ikke umiddelbart sættes sammen og da de to flows vi betragtede er de eneste mulige kan vi konkludere, at det fulde kommunikationsproblem ikke kan løses i tiden 1 ved blot at videresende meddelelser



Figur 1.1: *Sommerfuglenetværket.*



Figur 1.2: *De to partielle løsninger.*



Figur 1.3: *Løsning fundet ved brug af netværkskodning.*

gennem netværket. Det er naturligvis muligt blot at sende først som den ene partielle løsning og lade de kanter den ikke benytter være ubrugt og derefter sende som den anden partielle løsning. På denne måde får vi alt det ønskede data igennem, men vi skal bruge to tidsenheder på det. Dette er et eksempel på det, der kaldes routing, altså blot at videresende, og i praksis er dette som oftest dårligere end at anvende netværkskodning.
Det viser sig nemlig, at med netværkskodning kan problemet løses i tiden 1. Situationen er illustreret på figur 1.3. Som før lader vi s sende meddelelse a på kant 1 og meddelelse b på kant 2. Punkt v_1 videresender blot a til kant 3 og kant 5. På samme måde videresender v_2 meddelelse b til kant 4 og kant 7. Punktet v_3 modtager a fra kant 3 og b fra kant 4, denne videresender nu summen $a + b$ på kant 6. Punkt v_4 videresender dette til kant 8 og kant 9.

Modtager r_1 får altså meddelelse a på kant 5 og meddelelse $a + b$ på kant 8, ud fra dette kan r_1 finde b ved at addere det, der modtages på kant 5 og kant 8. Vi siger, at r_1 kan dekode meddelelsesmængden $\{a, b\}$. Det samme gør sig gældende for r_2.
Hvis et punkt, har mere end ét input kaldes det et indkodningspunkt, da det er i disse punkter, der potentielt skal ske noget andet end blot videresendelse. Dette er altså et eksempel på netværkskodning. Ordet kodning refererer til

indkodningsfunktioner, som givet input på et punkt bestemmer hvad det skal videresende og til dekodningsfunktioner i modtagerpunkter.

1.1 Netværkskodningsproblemer

Vi har nu introduceret netværkskodning uformelt, lad os introducere det mere formelt.
Lad $\mathcal{G} = (V, E)$ være en orienteret graf, med punktmængde V og kantmængde E. Vi tillader flere kanter mellem de samme punkter, og har derfor $E \subseteq V \times V \times \mathbb{Z}_+$, hvor det sidste heltal enumererer kanten, således at hvis der er to punkter med mere end en kant imellem sig, kan vi kende forskel på dem. Lad $\mathcal{S} = \{s_1, \ldots, s_S\} \subseteq V$ og $\mathcal{R} = \{r_1, \ldots, r_R\} \subseteq V$ være givet. Punkterne i $\mathcal{S}$ kaldes afsendere og punkterne i $\mathcal{R}$ kaldes modtagere. For et netværk har vi en mængde af beskeder $\mathcal{M} = \{M_1, \ldots, M_h\}$, som hver især kan tage alle værdier i $\mathcal{A}$, hvor $\mathcal{A}$ er et endeligt alfabet. Lad for alle afsendere s gælde at $\mathcal{X}(s) = \{X_1^{(s)}, \ldots, X_{\mu(s)}^{(s)}\} \subseteq \mathcal{M}$ er en mængde af $\mu(s)$ beskeder, som genereres i s. Vi kalder denne funktion for beskedfunktionen. Vi ønsker at tillade kommunikation mellem afsendere og modtagere, altså ønsker vi ved hjælp af netværket at replikere en delmængde af beskederne $\mathcal{X}(s)$ i et andet punkt r. Vi betragter altid kommunikation i et forsinkelsesfrit netværk, hvilket betyder, at vi altid kun har beskeder fra en afsendelse af gangen, og vi kan være sikre på, at der ikke kan komme til at løbe forskellig information på den samme kant fordi noget af informationen var forsinket. Lad for enhver modtager kravfunktionen $\mathcal{D}(r) = \{D_1^{(r)}, \ldots, D_{\nu(r)}^{(r)}\} \subseteq \mathcal{M}$ være en mængde af $\nu(r)$ beskeder, som kræves i dette punkt og som altså er de beskeder vi søger at reproducere i punktet.
Kanter angives med parenteser, så kant l fra u til v noteres $e = (u, v, l) \in E$ og alle kanter antages at være orienterede. Lad $ud(v)$ være de kanter, der går ud af et punkt v og lad for en kant $e = (u, v, l)$ gælde, at $ud(e) = ud(v)$. På samme måde er $ind(u)$ de kanter, der går ind i et punkt u og for en kant $e = (u, v, l)$ er $ind(e) = ind(u)$.
Vi antager generelt for netværk, at de er fri for kredse, hvilket betyder, at vi kan ordne kanterne på en sådan måde, at når der eksisterer en orienteret vej, hvor vi besøger kant e_1 før kant e_2, så er $e_1 \prec e_2$, hvilket kaldes en ancestral ordning.
Vi definerer en forbindelse c mellem en afsender og en modtager, som en triple $(s, r, \mathcal{X}(s, r)) \in \mathcal{S} \times \mathcal{R} \times \mathcal{P}_{\mathcal{X}(s)}$, hvor $\mathcal{P}_{\mathcal{X}(s)}$ er potensmængden af $\mathcal{X}(s)$. I en forbindelse $c = (s, r, \mathcal{X}(s, r))$ har vi, at $s = \text{afsender}(c) \in \mathcal{S}$ og $r = \text{modtager}(c) \in \mathcal{R}$. Af notationsmæssige årsager antager vi, at $\text{afsender}(c) \neq \text{modtager}(c)$.

Funktion $\mathcal{X}(s,r)$ beskriver hvilke beskeder vi ønsker at sende fra s til r. Vi har derfor også $\mathcal{D}(r) = \{\mathcal{X}(s,r) \mid s \in \mathcal{S}\}$.
Et punkt u kan sende information på en kant $e = (u,v,l)$ - det som sendes på kanten kaldes $Y(e)$. Ud over, at kende et evt. $\mathcal{X}(u)$ kan et punkt u observere $Y(e')$ for alle $e' \in ind(u)$. Generelt vil variablen $Y(e)$, som sendes på en kant $e \in ud(u)$ være en funktion af både $\mathcal{X}(u)$ og $Y(e')$, hvor $e' \in ind(u)$.
For alle kanter e i netværket har vi altså en variabel $Y(e)$, som tager værdier i $\mathcal{A}$. Relationen mellem variable i netværket er, at hvis vi besøger kanterne i ancestral orden, så har vi for hver kant $j = (u,v,l)$ en funktion f_j således, at

$$Y(j) = f_j\left((Y(i) \mid i \in ind(j)), (X_k^{(u)} \mid X_k^{(u)} \text{ genereres i } u)\right).$$

Hvis argumentet i f_j er tomt vil $Y(j)$ altid tage værdien 0. Funktionen f_j kaldes indkodningsfunktionen.
Hvis r er modtageren på nogle forbindelser er samlingen af $\kappa(r)$ beskeder $\mathcal{Z}(r) = \{Z_1^{(r)}, \ldots, Z_{\kappa(r)}^{(r)}\}$ de beskeder som eksisterer i r. For alle modtagere r har vi altså $\kappa(r)$ variable $Z_1^{(r)}, \ldots, Z_{\kappa(r)}^{(r)}$, som tager værdier i $\mathcal{A}$. Disse variabler er funktioner af de variabler, som genereres i r, og de beskeder, som kommer ind i r. For $j = 1, \ldots, \kappa(r)$ har vi en funktion $d_j^{(r)}$ således, at

$$Z_j^{(r)} = d_j^{(r)}\left((Y(i) \mid i \in ind(r)), (X_k \mid X_k \text{ genereres i } r)\right).$$

Et netværk $\mathcal{G} = (V,E)$ med en mængde af afsendere $\mathcal{S}$, en mængde af modtagere $\mathcal{R}$, en mængde af beskeder $\mathcal{M}$, en mængde af forbindelser $\mathcal{C}$, beskedfunktion $\mathcal{X}$ og kravfunktion $\mathcal{D}$ kaldes et netværkskodningsproblem. Vi vil angive et netværkskodningsproblem som et par $(\mathcal{G}, \mathcal{C})$, hvilket implicit giver alle ovenstående mængder og funktioner. Det siges at være løseligt, hvis der eksisterer et ikke-trivielt alfabet $\mathcal{A}$ og en mængde af valg af indkodningsfunktioner f_j og dekodningsfunktioner $d_j^{(r)}$ således, at

$$\mathcal{Z}(r) = \{Z_1^{(r)}, \ldots, Z_{\kappa(r)}^{(r)}\} \supseteq \mathcal{D}(r)$$

holder for alle $r \in \mathcal{R}$. På tilsvarende vis siger vi, at en forbindelse $c = (s, r, \mathcal{X}(s,r))$ er succesfuld hvis en kopi af $\mathcal{X}(s,r)$ er en delmængde af $\mathcal{Z}(r)$. Alle forbindelser skal være succesfulde for at netværkskodningsproblemet er løseligt.
Hvis $\mathcal{D}(r) = \mathcal{M}$ for alle $r \in \mathcal{R}$ (altså alle modtagere vil have alle beskeder), så kalder vi netværkskodningsproblemet for et multicast netværkskodningsproblem.

Hvis alfabetet er et endeligt legeme $\mathbb{F}_q$ og hvis alle indkodningsfunktioner f_j såvel som alle dekodningsfunktioner $d_j^{(r)}$ er lineære over $\mathbb{F}_q$, så kalder vi det lineær netværkskodning, hvilket er en delmængde af generel netværkskodning. På tilsvarende vis snakker vi om et lineært netværkskodningsproblem.
Lad os nu definere variablen $Y(e)$ i et acyklisk netværk, der er lineært i $\mathbb{F}_q$.

Definition 1.1:
Lad $\mathcal{G} = (V, E)$ være et acyklisk kommunikationsnetværk. Vi siger, at $\mathcal{G}$ er et $\mathbb{F}_q$-lineært netværk, hvis for alle kanter $j = (u, v, l) \in E$ vi har, at $Y(j)$ opfylder, at

$$Y(j) = \sum_{i \in ind(j)} f_{i,j} Y(i) + \sum_{\substack{u \in \mathcal{S} \\ X_i \in \mathcal{X}(u)}} a_{i,j} X_i,$$

hvor $f_{i,j} \in \mathbb{F}_q$ og $a_{i,j} \in \mathbb{F}_q$.

Det gælder ydermere for lineær netværkskodning, at

$$Z_j^{(r)} = \sum_{i \in ind(r)} b_{i,j}^{(r)} Y(i) + \sum_{\substack{r \in \mathcal{S} \\ X_i \in \mathcal{X}(r)}} \widetilde{b}_{i,j}^{(r)} X_i, \tag{1.3}$$

hvor $b_{i,j}^{(r)}, \widetilde{b}_{i,j}^{(r)} \in \mathbb{F}_q$. Hvis vi antager, at $\mathcal{S}$ og $\mathcal{R}$ er disjunkte får vi, at

$$\sum_{\substack{r \in \mathcal{S} \\ X_i \in \mathcal{X}(r)}} \widetilde{b}_{i,j}^{(r)} X_i = 0$$

og dermed udgår i (1.3).

I et netværk er et flow mellem s og r er en mængde af t kantdisjunkte veje fra s til r. Vi betragter et flow som en samling af kanter, så givet et flow F mener vi med $j \in F$, at j er en kant, der optræder i F. Dette kan naturligt udvides til flow mellem mængder af punkter, idet vi igen definerer det som en mængde af kantdisjukte veje mellem mængderne af punkter.

Vi vil nu forsøge at relatere løseligheden af et netværkskodningsproblem til størrelsen af flows mellem afsendere og modtagere. Vi indfører derfor min-cut maks-flow sætningen.

Sætning 1.2 (Min-cut max-flow):
I ethvert netværk, er det maksimale flow mellem to punkter lig med det minimale cut mellem samme to punkter.

Bevis findes i [3].
Det kan forklares med følgende. Et cut mellem s og r i en graf $\mathcal{G}$, er en kantmængde C, som hvis den fjernes vil opdele grafen $\mathcal{G}$ i to delgrafer $\mathcal{G}_1, \mathcal{G}_2 \subset \mathcal{G}$, således, at $s \in \mathcal{G}_1$ og $r \in \mathcal{G}_2$.
Det betyder altså, at det maksimale antal kantdisjunkte veje, som kan eksistere mellem s og r nødvendigvis må være lig med det mindste antal kanter, som skal fjernes for at skille s og r fuldstændigt, altså det minimale cut.

Sætning 1.3:
Det er ikke muligt at lave et multicast af størrelse h, hvis der eksisterer en modtager $r \in \mathcal{R}$ således, at det maksimale flow mellem s og r har størrelse mindre end h.

Bevis:
Betragt et multicast fra afsender s til modtagergruppen R af h beskeder. Antag, at det maksimale flow er mindre end h fra s til en modtager r. Dette betyder ifølge min-cut maks-flow sætningen 1.2, at det minimale cut mellem s og r også er mindre end h. Lad os kalde værdien af det minimale cut for k. Vi kan altså forestille os netværket som værende to dele forbundet af k kanter, hvor vi i den ene del d_1 har s og den anden del d_2 har r.
Vi kan se at s kan sende q^h kombinationer af tegn, hvor q er alfabetstørrelsen, og vi kan se at det kun er muligt at sende q^k kombinationer af tegn fra d_1 til d_2, og da $q^h > q^k$ må det gælde, at nogle af de kombinationer der bliver sendt fra s afbilleder til den samme kombination, der sendes mellem d_1 og d_2 og derfor er det ikke muligt for modtageren r at dekode til den rigtige kombination.

■

Det viser sig ydermere, at i tilfælde med multicast af h beskeder er eksistensen af et flow af størrelse h for alle modtagere faktisk tilstrækkeligt for, at der eksisterer en løsning og ydermere kan vi nøjes med lineær netværkskodning. Dette vil vi diskutere i det følgende afsnit og bevise i kapitel 4. I ethvert netværk gælder ydermere, at hvis der kun er én modtager og der eksisterer et stort nok flow for denne modtager er routing faktisk nok til at løse netværkskodningsproblemet. Dette skyldes, at vi definerer, at vejene i et flow er kantdisjunkte.

Derfor er der altså nok separate veje mellem alle afsendere og vores modtager til at sende hver enkelt besked direkte. Der er altså ikke grund til at ændre på disse undervejs, og de kan blot sendes videre hele vejen gennem netværket.
Lad os betragte følgende sætning om relationen mellem størrelsen af et flow mellem to punkter og successen af en forbindelse mellem samme to punkter.

Sætning 1.4:
Betragt en forbindelse $c = (s, r, \mathcal{X}(s, r))$. Hvis ikke det gælder, at $|\mathcal{X}(s, r)|$ er mindre end eller lig med det maksimale flow mellem s og r kan c ikke være succesfuld.

Bevis:
En forbindelse $c = (s, r, \mathcal{X}(s, r))$ er succesfuld, hvis $\mathcal{X}(s, r)$ er en delmængde af $\mathcal{Z}(r)$.
Når vi betragter en enkelt forbindelse svarer dette til, at betragte et multicast netværkskodningsproblem, hvor der er en afsender, en modtager og $h = |\mathcal{X}(s, r)|$ beskeder. Forbindelsen er derfor succesfuld, når netværket er løseligt.
Vi ved fra sætning 1.3 at et multicast netværkskodningsproblem kan løses kun hvis der eksisterer et flow af størrelse $h = |\mathcal{X}(s, r)|$. Det betyder altså, at hvis $|\mathcal{X}(s, r)|$ er større end det maksimale flow mellem s og r kan netværkskodningsproblemet ikke løses og forbindelsen ikke være succesfuld.

■

Vi vil nu udvide fra en forbindelse til et helt netværkskodningsproblem.

Sætning 1.5:
Betragt et netværkskodningsproblem $(\mathcal{G}, \mathcal{C})$. Hvis ikke det for enhver forbindelse $c = (s, r, \mathcal{X}(s, r)) \in \mathcal{C}$ gælder, at $|\mathcal{X}(s, r)|$ er mindre end eller lig med det maksimale flow mellem s og r, så kan netværkskodningsproblemet ikke være løseligt.

Bevis:
Et netværk er løseligt kun hvis alle dets forbindelser er succesfulde, og for, at enhver af disse forbindelser $c = (s, r, \mathcal{X}(s, r))$ er succesfulde skal de ifølge sætning 1.4 opfylde at $|\mathcal{X}(s, r)|$ er mindre end eller lig med det maksimale flow mellem s og r.

■

1.2 Lineær netværkskodning for multicast netværkskodningsproblemer

Lad os nu dykke dybere ind i begrebet lineær netværkskodning og introducere dette formelt. Vi koncentrerer os om et multicast netværkskodningsproblem med h beskeder.
Lad A være en $h \times |E|$-matrix således, at

$$A_{i,j} = \begin{cases} a_{i,j} & \text{hvis } j \in ud(s), \text{ når } X_i \in \mathcal{X}(s) \\ 0 & \text{ellers.} \end{cases}$$

Lad F være en $|E| \times |E|$-matrix således, at

$$F_{i,j} = \begin{cases} f_{i,j} & \text{hvis } j \in ud(i) \\ 0 & \text{ellers.} \end{cases}$$

For $r \in \mathcal{R}$ lad $B^{(r)}$ være en $|E| \times h$-matrix således, at

$$B^{(r)}_{i,j} = \begin{cases} b^{(r)}_{i,j} & \text{hvis } i \in ind(r) \\ 0 & \text{ellers.} \end{cases}$$

Sætning 1.6:
For ethvert ikke-negativt heltal n er den (i,j)'te indgang i F^n givet ved

$$\sum_{\substack{(i=j_0,j_1,\ldots,j_n=j) \\ \text{er en vej i } \mathcal{G}}} f_{i=j_0,j_1} f_{j_1,j_2} \cdots f_{j_{n-1},j_n=j}.$$

Bevis:
Dette kan vises ved induktion på tallet n.
For $n = 1$ har vi for F^1, at den (i,j)'te indgang er givet ved

$$\sum_{\substack{(i,j) \\ \text{er en vej i } \mathcal{G}}} f_{i,j} = \begin{cases} f_{i,j} & \text{hvis } j \in ud(i) \\ 0 & \text{ellers.} \end{cases}$$

Antag, at for $n = k$ har vi for F^k, at den (i,j)'te indgang er givet ved

$$\sum_{\substack{(i=j_0,j_1,\ldots,j_k=j) \\ \text{er en vej i } \mathcal{G}}} f_{i=j_0,j_1} f_{j_1,j_2} \cdots f_{j_{k-1},j_k=j}.$$

For $n = k + 1$ har vi nu, at

$$F^{k+1} = F^k F^1$$

Hvis vi kalder indgangene i F^k for $\alpha_{i,j}$ og indgangene i F^1 for $\beta_{i,j}$ har vi

$$\begin{aligned} F^{k+1} = F^k F^1 &= \left[\sum_{l=1}^{|E|} \alpha_{i,l}\beta_{l,j} \right] \\ &= \left[\sum_{l=1}^{|E|} \left(\sum_{\substack{(i=j_0,j_1,\ldots,j_k=l) \\ \text{er en vej i } \mathcal{G}}} f_{i=j_0,j_1} f_{j_1,j_2} \cdots f_{j_{k-1},j_k=l} \sum_{\substack{(l,j) \\ \text{er en vej i } \mathcal{G}}} f_{l,j} \right) \right] \end{aligned}$$

For hver værdi af l vil vi enten have, at begge de inderste summer har en værdi (og der altså findes en vej) eller, at den ene og/eller den anden sum er 0. Derfor får vi, at

$$F^{k+1} = \left[\sum_{\substack{(i=j_0,j_1,\ldots,j_k=l,j_{k+1}=j) \\ \text{er en vej i } \mathcal{G}}} f_{i=j_0,j_1} f_{j_1,j_2} \cdots f_{j_{k-1},j_k=l} f_{j_k=l,j=j_{k+1}} \right]$$

Dette viser sætningen generelt, og den (i, j)'te indgang i F^n er altså som ønsket.

■

Matricen F^n indeholder altså information om alle veje af længde n. Da vi har antaget, at den orienterede graf er acyklisk og endelig har vi derfor, at F^N er lig med 0 matricen for et stort nok $N \in \mathbb{N}$, derfor indeholder matricen $(I + F + \cdots + F^{N-1})$ information om alle mulige veje i netværket. Hvis vi ganger A på dette, så vi får matricen $A(I + F + \cdots + F^{N-1})$, så vil denne indeholde information om hvor alle beskeder kan bevæge sig hen i alle antal skridt.
Vi konkluderer, at hvis vi giver værdier til alle $a_{i,j}$ og $f_{i,j}$ så har vi, at

$$(Y(1), \ldots Y(|E|)) = (X_1, \ldots, X_h)A(I + F + \cdots + F^{N-1})$$

er opfyldt. Hvis vi giver værdier til alle $b_{i,j}^{(r)}$ så giver dekodningsfunktionen hos modtager r, at

$$(Y(1), \ldots Y(|E|))B^{(r)} = (X_1, \ldots, X_h)A(I + F + \cdots + F^{N-1})B^{(r)}$$

eller

$$(Y(1),\dots Y(|E|))B^{(r)} = (X_1,\dots,X_h)M^{(r)},$$

hvor

$$\begin{aligned} M^{(r)} &= A(I+F+\cdots+F^{N-1})B^{(r)} \\ &= A(I-F)^{-1}B^{(r)} \end{aligned}$$

kaldes transfermatricen for r. Vi bruger det faktum, at $(I-F)(I+F+\cdots+F^{N-1}) = I$, hvilket ses af følgende sætning.

Sætning 1.7:
For matricen F, som beskrevet overfor gælder, at $(I-F)(I+F+\cdots+F^{N-1}) = I$

Bevis:
Hvis vi ganger de to parenteser ud får vi følgende teleskopsum,

$$\begin{aligned} (I-F)(I+F+\cdots+F^{N-1}) &= (I-F)+(F-F^2)+(F^2-F^3)+\cdots+(F^{N-1}-F^N) \\ &= I-F^N \\ &= I, \end{aligned}$$

idet $F^N = 0$.

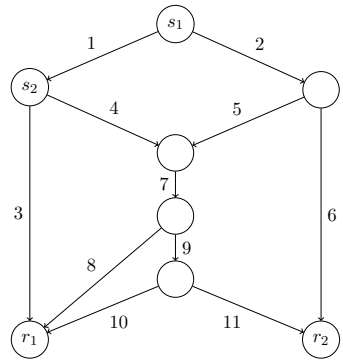
■

Lad os nu betragte et eksempel for at forstå hvordan transfermatricerne relaterer sig til netværkets topologi.

Eksempel 1.8:
Betragt netværket i figur 1.4. Vi ønsker følgende forbindelser $\mathcal{C} = \{(s_1,r_1,\{X_1^{(s_1)}\}),(s_1,r_2,\{X_2^{(s_1)}\}),(s_2,r_1,\{X_1^{(s_2)}\}),(s_2,r_2,\{X_2^{(s_2)}\})\}$.
Lad os nu lave de tilhørende matricer A, F, $B^{(r_1)}$ og $B^{(r_2)}$.

Matricen A er en 2×11-matrix

$$A = \begin{bmatrix} a_{1,1} & a_{1,2} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & a_{2,3} & a_{2,4} & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix},$$



Figur 1.4: *Der genereres en besked $X_1^{(s_1)}$ i s_1 og en besked $X_2^{(s_2)}$ i s_2.*

matricen F er en 11×11-matrix

$$F = \begin{bmatrix} 0 & 0 & f_{1,3} & f_{1,4} & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & f_{2,5} & f_{2,6} & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & f_{4,7} & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & f_{5,7} & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & f_{7,8} & f_{7,9} & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & f_{9,10} & f_{9,11} \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

og slutteligt er $B^{(r)}$-matricerne 11×2-matricer

$$B^{(r_1)} = \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ b_{3,1} & b_{3,2} \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ b_{8,1} & b_{8,2} \\ 0 & 0 \\ b_{10,1} & b_{10,2} \\ 0 & 0 \end{bmatrix} \quad B^{(r_2)} = \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ b_{6,1} & b_{6,2} \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ b_{11,1} & b_{11,2} \end{bmatrix}.$$

Vi kan nu regne transfermatricerne $M^{(r_1)}$ og $M^{(r_2)}$

$$M^{(r_1)} = \begin{bmatrix} M_{1,1}^{(r_1)} & M_{1,2}^{(r_1)} \\ M_{2,1}^{(r_1)} & M_{2,2}^{(r_1)} \end{bmatrix}$$

$$\begin{aligned}
M_{1,1}^{(r_1)} =& a_{1,1}f_{1,3}b_{3,1} + \left(a_{1,1}f_{1,4}f_{4,7}f_{7,8} + a_{1,2}f_{2,5}f_{5,7}f_{7,8}\right)b_{8,1} \\
&+ \left(a_{1,1}f_{1,4}f_{4,7}f_{7,9}f_{9,10} + a_{1,2}f_{2,5}f_{5,7}f_{7,9}f_{9,10}\right)b_{10,1} \\
M_{1,2}^{(r_1)} =& a_{1,1}f_{1,3}b_{3,2} + \left(a_{1,1}f_{1,4}f_{4,7}f_{7,8} + a_{1,2}f_{2,5}f_{5,7}f_{7,8}\right)b_{8,2} \\
&+ \left(a_{1,1}f_{1,4}f_{4,7}f_{7,9}f_{9,10} + a_{1,2}f_{2,5}f_{5,7}f_{7,9}f_{9,10}\right)b_{10,2} \\
M_{2,1}^{(r_1)} =& a_{2,4}f_{4,7}f_{7,9}f_{9,10}b_{10,1} + a_{2,4}f_{4,7}f_{7,8}b_{8,1} + a_{2,3}b_{3,1} \\
M_{2,2}^{(r_1)} =& a_{2,4}f_{4,7}f_{7,9}f_{9,10}b_{10,2} + a_{2,4}f_{4,7}f_{7,8}b_{8,2} + a_{2,3}b_{3,2}
\end{aligned}$$

$$M^{(r_2)} = \begin{bmatrix} M_{1,1}^{(r_2)} & M_{1,2}^{(r_2)} \\ M_{2,1}^{(r_2)} & M_{2,2}^{(r_2)} \end{bmatrix}$$

$$\begin{aligned}
M_{1,1}^{(r_2)} =& a_{1,2}f_{2,6}b_{6,1} + \left(a_{1,1}f_{1,4}f_{4,7}f_{7,9}f_{9,11} + a_{1,2}f_{2,5}f_{5,7}f_{7,9}f_{9,11}\right)b_{11,1} \\
M_{1,2}^{(r_2)} =& a_{1,2}f_{2,6}b_{6,2} + b_{11,2}a_{1,1}f_{1,4}f_{4,7}f_{7,9}f_{9,11} + b_{11,2}a_{1,2}f_{2,5}f_{5,7}f_{7,9}f_{9,11} \\
M_{2,1}^{(r_2)} =& a_{2,4}f_{4,7}f_{7,9}f_{9,11}b_{11,1} \\
M_{2,2}^{(r_2)} =& a_{2,4}f_{4,7}f_{7,9}f_{9,11}b_{11,2}
\end{aligned}$$

Vi kan nu se, at alle leddene i indgangene i $M^{(r)}$-matricerne består af et enkelt a nogle f'er og et enkelt b. Hvis vi for eksempel ser på $M_{2,2}^{(r_2)}$ kan vi se at det svarer til den eneste vej der er fra s_2 til r_2. Da $M^{(r)}$-matricerne er besked $\times$ besked matricer har vi, at indgangene i matricerne repræsentere vejen startende med en besked og sluttende med en besked. Det vil altså sige, at $M_{i,j}^{(r)}$ repræsenterer vejen fra stedet hvor besked i genereres og til r, som ønsker besked j. Det er derfor kun i diagonalen, at vi starter og slutter med samme besked.

Med denne nye viden om transfermatricerne kan vi nu betragte hvordan værditildelingen skal kunne laves for variablerne for at netværket er løseligt.

Da vi har, at dekodningsfunktionen hos modtager r giver, at

$$(Y(1), \ldots Y(|E|))B^{(r)} = (X_1, \ldots, X_h)M^{(r)},$$

må det gælde, at vi ønsker at have $M^{(r)} = I$.
Hvis vi slækker lidt på kravene for succesfuld ind- og dekodning kan vi i stedet for at kræve, at $M^{(r)} = I$ for alle $r \in \mathcal{R}$ blot kræve, at $M^{(r)}$ har fuld rang for alle $r \in \mathcal{R}$. Vi kræver altså, at

$$\prod_{r \in \mathcal{R}} \det(M^{(r)}) \neq 0 \tag{1.4}$$

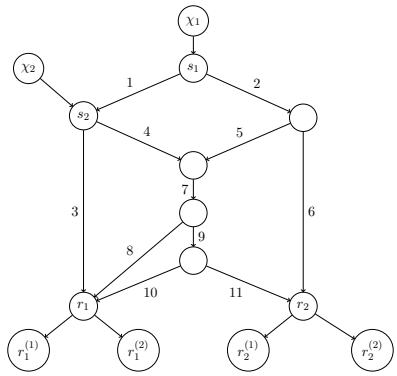
og kalder dette produkt for transferpolynomiet.
Givet koefficienter $a_{i,j}$, $f_{i,j}$ og $b_{i,j}^{(r)}$ således, at (1.4) er opfyldt er det klart, at man kan ændre værdien af $b_{i,j}^{(r)}$ således, at $M^{(r_1)} = \cdots = M^{(r_{|\mathcal{R}|})} = I$ er opfyldt.
Det er vist i [8], at $\det(M^{(r)})$ enten er lig med $\det(E^{(r)})$ eller $-\det(E^{(r)})$, hvor

$$E^{(r)} = \begin{bmatrix} A & 0 \\ I - F & B^{(r)} \end{bmatrix}$$

kaldes Edmond-matricen for r. Fordelen ved Edmond-matricen frem for transfermatricen er, at vi ikke behøver at finde et tal N så $F^N = 0$. For at bestemme hvorvidt $\prod_{r \in \mathcal{R}} \det(M^{(r)}) \neq 0$ er det nok at afgøre hvorvidt $\prod_{r \in \mathcal{R}} \det(E^{(r)}) \neq 0$.
Det er ikke overraskende, at determinanten i transfermatricen har en topologisk mening. Denne forklares lettest med et eksempel.

Eksempel 1.9:
Givet et multicast netværkskodningsproblem starter vi med at udvide grafen $\mathcal{G} = (E, V)$ som følger (se figur 1.5).
Først tilføjer vi h nye punkter $\chi_1, \ldots, \chi_h$ til V. Så tilføjer vi til E for hvert $i = 1, \ldots, h$ en kant fra χ_i til det punkt hvor X_i genereres. For hver modtager $r \in \mathcal{R}$ tilføjer vi h nye punkter $r^{(1)}, \ldots, r^{(h)}$. Og endelig tilføjer vi en kant fra r til ethvert punkt $r^{(i)}$, $i = 1, \ldots, h$. Vi betragter nu et flow mellem $\{\chi_1, \ldots, \chi_h\}$ til $\{r^{(1)}, \ldots, r^{(h)}\}$, som er en mængde af kantdisjunkte veje. En vej i dette flow svarer på en naturlig måde til et produkt af et $a_{i,j}$, nogle $f_{i,j}$'er og et $b_{i,j}$.

Hvis vi bruger sætning 1.6 og den efterfølgende diskussion ser vi, at den (i,j)'te indgang i $M^{(r)}$ indeholder information om alle veje fra χ_i til $r^{(j)}$. Det kan ses,



Figur 1.5: *Udvidelsen af netværket i figur 1.4.*

at der er en en-til-en korrespondance mellem flows mellem $\{\chi_1, \ldots, \chi_h\}$ og $\{r^{(1)}, \ldots, r^{(h)}\}$ i den udvidede graf, og det monomielle udtryk, der opstår i $\det(M^{(r)})$. Ethvert monomielt udtryk optræder med koefficienten 1 eller -1. Hvis vi betragter produktet $\prod_{r\in\mathcal{R}} \det(M^{(r)})$ er koefficienterne ikke længere blot 1'ere og -1'ere men medlemmer af $\mathbb{F}_p$, hvor p er karakteristikken for legemet $\mathbb{F}_q$ (altså $\mathbb{F}_{p^m}$) som betragtes. Faktisk kan termerne gå ud med hinanden alt efter karakteristikken af legemet. Bemærk dog, at det altid kun kan gælde, at $\det(M^{(r)}) \neq 0$ for alle $r \in \mathcal{R}$ hvis og kun hvis $\prod_{r\in\mathcal{R}} \det(M^{(r)}) \neq 0$. Hvilket giver os følgende sætning.

Sætning 1.10:
Transferpolynomiet er ikke-nul hvis og kun hvis der eksisterer et flow af størrelse h fra $\mathcal{S}$ til alle modtagerne $r \in \mathcal{R}$.

1.3 Algoritme til løsning af multicast netværkskodningsproblemer

Dette afsnit er baseret på en artikel af Geil og Thomsen [7].
Givet et multicast netværkskodningsproblem, som kan løses, ønsker vi at have en algoritme, som nemt og hurtigt kan finde en løsning til problemet. Vi ved, at Ford-Fulkerson algoritmen effektivt kan finde flows i et netværk [1]. Vi kalder en mængde af flows for et flowsystem og det viser sig, at givet flowsystem af størrelse h, så findes der en algoritme, som kan finde en løsning over $\mathbb{F}_q$ for

ethvert $q \geq |\mathcal{R}|$. Denne algoritme, som blev lavet af Jaggi et al. [10], bruger globale kodningsvektorer, som er givet ved følgende definition.

Definition 1.11 (Globale kodningsvektorer):
Antag, at koefficienter $a_{i,j}$, $f_{i,j}$ er valgt for et netværkskodningsproblem. For enhver kant j definerer vi den tilsvarende globale kodningsvektor til at være $c_g(j) = (c_1, \ldots, c_h)$ hvis $Y(j) = c_1X_1 + \cdots + c_hX_h$.

Det er klart, at hvis det for enhver modtager r, gælder, at de globale kodningsvektorer af $ind(r)$ udspænder hele $\mathbb{F}_q^h$, så må indkodningskoefficienterne være del af en løsning til netværkskodningsproblemet. Det eneste vi mangler, for at have en fuldstændig løsning til problemet er at vælge $b_{i,j}^{(r)}$'erne, hvilket vi kan gøre ved hjælp af ligningerne fra (1.3) på side 6.
Algoritmen fungerer ved først at modificere netværkskodningsproblemet en smule. Først tilføjes et punkt s' til netværket. Derefter bliver der for hvert $s_i \in \mathcal{S}$ tilføjet $v(s_i)$ kanter fra s' til s_i. Her er $v(s_i)$ antallet af beskeder, som genereres i s_i. Alt i alt tilføjes altså h kanter. Derefter antages det, at alle beskeder genereres i s'. Beskriv ved $\mathcal{F} = (F_1, \ldots, F_{|\mathcal{R}|})$ et flowsystem, her er F_i flowet fra s' til modtager r_i. Algoritmen virker på det modificerede netværkskodningsproblem og initialiserer på følgende måde:

- Hvis (i, j) ikke er en del af nogen vej i flowsystemet sætter vi $f_{i,j} = 0$
- Indkodningsfunktioner i punktet s' vælges på en sådan måde, at de globale kodningsvektorer til de h tilføjede kanter er

$$(1, 0, \ldots, 0), (0, 1, 0, \ldots, 0), \ldots, (0, \ldots, 0, 1) \tag{1.5}$$

  Denne information bliver ikke gemt, idet det ikke er relevant for det oprindelige problem.

Bemærk, at de h kanter er et cut i flowet til hver modtager. Vi initialiserer $C_1 = \cdots = C_{|\mathcal{R}|}$ til at være den ordnede mængde af de h tilføjede kanter og $B_1, \ldots, B_{|\mathcal{R}|}$ til at være den tilsvarende ordnede mængde af globale kodningsvektorer (som svarer til (1.5)).
Hoveddelen af algoritmen er et loop hvori de ordnede mængder $C_1, \ldots, C_{|\mathcal{R}|}$ bliver opdateret og hvor kodningskoefficienterne, som er relateret til opdateringen vælges således, at for $l = 1, \ldots, |\mathcal{R}|$ gælder følgende loop invariant:

- C_l er et cut i F_l
- Den ordnede mængde B_l af globale kodningsvektorer svarende til C_l udspænder hele $\mathbb{F}_q^h$.

Opdateringerne gøres således, at det til sidst (efter endeligt mange skridt) gælder, at $C_l = ind(r_l) \cap F_l$, for $l = 1, \ldots, |\mathcal{R}|$. Dette betyder, at ved den anden loop invariant har enhver modtager nok information til at udregne passende $b_{i,j}^{(r)}$'er ved hjælp af simpel lineær algebra.
Opdateringen udføres som følger:
Vi besøger kanterne $j \in \mathcal{F}$ i rækkefølgen givet ved ancestral ordning. Idet vi kigger på en kant j interessere vi os for modtagere r_l således, at $j \in F_l$. Vi tilføjer j til C_l og fjerner fra C_l dens forgænger i F_l. Det betyder, at vi fjerner den unikke kant $i \in C_l \cap ind(j)$ for hvilken (i, j) er en del af en vej i F_l. Lad $l_1, \ldots, l_k$ være værdier af l for hvilke C_l bliver opdateret og lad $i_1, \ldots, i_k$ (muligvis er $i_s = i_t$ for nogle $s \neq t$) være de kanter, som fjernes fra $C_{l_1}, \ldots, C_{l_k}$. Opgaven er nu, at vælge koefficienter $f_{i_1,j}, \ldots, f_{i_k,j}$ på en sådan måde, at alle mængder af globale kodningsvektorer $B_{l_1}, \ldots, B_{l_k}$ for $C_{l_1}, \ldots, C_{l_k}$ udspænder hele $\mathbb{F}_q^h$. Hvis kodningskoefficienterne vælges tilfældigt i et stort nok legeme er sandsynligheden for succes positiv, hvilket garanteres af følgende lemma.

Lemma 1.12:
Givet en basis $\{\vec{b}_1, \ldots, \vec{b}_h\}$ for $\mathbb{F}_q^h$ og $\vec{c} \in \mathbb{F}_q^h$ er der præcis ét valg af $a \in \mathbb{F}_q$ således, at

$$\vec{c} + a\vec{b}_h \in Span_{\mathbb{F}_q}\{\vec{b}_1, \ldots, \vec{b}_{h-1}\}. \tag{1.6}$$

Bevis:
Skriv $\vec{c}$ som en linearkombination $\vec{c} = c_1\vec{b}_1 + \cdots + c_h\vec{b}_h$. Det eneste a for hvilken (1.6) holder er $a = -c_h$.

■

For ethvert i_t kan vi nu benytte lemma 1.12 med basis B_{l_t} (som den er før opdateringen), med $\vec{b}_h = c_g(i_t)$ og med

$$\vec{c} = \sum_{l \in \{i_1, \ldots, i_k\} \setminus \{i_t\}} f_{l,j} c_g(l).$$

Lad $k' = |\{i_1, \ldots, i_k\}| = |ind(j) \cap \mathcal{F}|$.
Vi har nu, at $q^{k'}$ er alle mulige kombinationer af valg på alle kanter i $ind(j)$,

altså kombinationer af valg af $f_{l,j}$. Hvis vi vælger funktioner til $k'-1$ af kanterne vil der ifølge lemmaet være ét valg på den sidste kant, som ikke virker, for hver enkelt modtager. Det betyder, at der er $k \cdot q^{k'-1}$ kombinationer, der ikke virker. Dog har vi i disse kombinationer medtaget nulløsningen (alle funktioner er nulfunktionen, $f_{l,j} = 0$ for alle l) k gange, da denne ikke er en løsning for nogen af modtagerne. Denne løsning skal kun medtages en gang, og derfor er der reelt højst $k \cdot q^{k'-1} - (k-1)$ kombinationer, som ikke virker.
Vi kan nu opstille sandsynligheden for at vælge en rigtig løsning således,

$$\frac{q^{k'} - (k \cdot q^{k'-1} - (k-1))}{q^{k'}} = 1 - \frac{k}{q} - \frac{k-1}{q^{k'}}$$

Denne er positivt for $q \geq k$ og da vi har $k \leq |\mathcal{R}|$ har vi vist, at sandsynligheden for at vælge rigtigt i et skridt af algoritmen er positiv for $q \geq |\mathcal{R}|$. Det betyder, at sandsynligheden for, at algoritmen vælger rigtigt i alle skridt også er positiv. For et stort nok q vil sandsynligheden endda gå mod 1.
For et løseligt netværkskodningsproblem er $\mathbb{F}_q$ altså nok, hvis $q \geq |\mathcal{R}|$.
Dette vil altså samtidig sige, at hvis vi for et multicast netværkskodningsproblem vælger lineære funktioner i alle indkodningspunkter fuldstændig tilfældigt, så er sandsynligheden for, at dette går godt også positiv. Dette svarer til, at vælge tilfældigt i alle indkodningspunkter i flowet og i alle andre punkter, så det er altså lidt udvidet i forhold til hvad algoritmen gør. Det skal bemærkes, at de punkter, som tilhører flowet muligvis har ekstra kanter ind ifht flowet og altså dermed muligvis noget ekstra støj. Vi kan dog stadig bruge samme argumentation i denne situation og stadig bruge lemma 1.12 idet vi nu blot har en større basis. Der er altså stadig et valg, som ikke fungerer, i hvert af de vigtige indkodningspunkter, det vil sige punkter, som er i flowet. Derfor er sandsynligheden for at lave en løsning ved tilfældigt at vælge funktioner til alle indkodningspunkter positiv, idet det svarer til valgene i algoritmen.

Vi har således introduceret netværkskodning og netværkskodningsproblemer generelt og lineær netværkskodning og multicast netværkskodningsproblemer i særdeleshed. Lad os nu for en stund forlade emnet netværkskodning og i stedet benytte lejligheden til at dykke ind i abstrakt algebra.

Algoritme 1: Algoritme til randomiseret generering af indkodningsfunktioner, som beskrevet af Jaggi et. al

Input: Et løseligt multicast (acyklisk) netværkskodningsproblem med $\mathcal{G} = (V, E)$. Et legeme $\mathbb{F}_q$ med $q \geq |\mathcal{R}|$. En ancestral ordning af E.

Output: Indkodningskoefficienter $a_{i,j}$, $f_{i,j}$.

Initialisering

Lad $V' := V \cup \{s'\}$ og $E' := E \cup \{e_1, \ldots, e_h\}$ hvor de nye kanter er som beskrevet i teksten

Find et flowsystem $\mathcal{F} = \{F_1, \ldots, F_{|\mathcal{R}|}\}$ i $\mathcal{G}' := (V', E')$ af størrelse h fra s' til modtagerne

for $j \in E$ *og* $i = 1, \ldots, h$ **do**
  if $j \in ud(e_i)$ *og* $(e_i, j) \notin \mathcal{F}$ **then** $a_{i,j} = 0$
end

for $(i, j) \in E \times E$, *hvor* $i \in ind(j)$ *og* $(i, j) \notin \mathcal{F}$ **do** $f_{i,j} = 0$

for $l = 1, \ldots, |\mathcal{R}|$ **do**
  $C_l := (e_1, \ldots, e_h)$
  $B_l := ((1, 0, \ldots, 0), (0, 1, 0, \ldots, 0), \ldots, (0, \ldots, 0, 1))$
end

Opdatering

for $j \in \mathcal{F}$ *efter den nedarvede orden* **do**
  Lad $(r_{l_1}, \ldots, r_{l_k})$ være de modtagere, der bruger j i $\mathcal{F}$
  Lad $(i_1, \ldots, i_k)$ være de tilsvarende forgængere til j i $\mathcal{F}$
  for $v = 1, \ldots, k$ **do**
    $C_{l_v} := (C_{l_v} \backslash \{i_v\}) \cup \{j\}$
  end
  Vælg indkodningskoefficienter (notation med hensyn til det udvidede netværk) $(f_{i_1,j}, \ldots, f_{i_k,j})$ tilfældigt indtil B_{l_v} udspænder $\mathbb{F}_q^h$ for alle $v = 1, \ldots, k$
  for $v = 1, \ldots, k$ **do**
    if i_v *er lig med* e_k *for et* $k \in \{1, \ldots, h\}$ **then**
      omdøb $f_{i_v,j}$ til $a_{k,j}$
    end
  end
end

return *alle* $a_{i,j}, f_{i,j}$ *til hvilke der er blevet tildelt værdier*

Kapitel 2

Gröbnerbaser

Vi vil i det følgende introducere en lang række sætninger og definitioner, som har at gøre med Gröbnerbasisteori. Denne teori vil vi senere anvende i arbejdet med udvælgelse af funktioner til netværkskodningen. Afsnittet er skrevet ud fra en bog af Cox et al. [2].

2.1 Monomier og polynomier

Lad os starte med at definere monomier og polynomier.

Definition 2.1 (Monomium):
Et monomium i variablerne $x_1, x_2, \ldots, x_n$ er et produkt på formen

$$x_1^{\alpha_1} x_2^{\alpha_2} \cdots x_n^{\alpha_n}$$

hvor $\alpha_1, \alpha_2, \ldots, \alpha_n$ er ikke-negative heltal. Den totale grad af monomiet er summen $\alpha_1 + \alpha_2 + \ldots + \alpha_n$

Vi kan simplificere notation af monomier ved at lade $\alpha = (\alpha_1, \alpha_2, \ldots, \alpha_n)$ være en n-tuple af ikke-negative tal og definere, at

$$x^\alpha = x_1^{\alpha_1} x_2^{\alpha_2} \cdots x_n^{\alpha_n}.$$

I praksis vil vi, når vi arbejder med tre eller færre variabler, kalde disse for x, y og z i stedet for x_1, x_2 og x_3. Vi kan nu definere et polynomium.

Definition 2.2 (Polynomium):
Et polynomium i $x_1, x_2, \ldots, x_n$ med koefficienter i k, hvor k er et endeligt legeme, er en endelig sum på formen

$$\sum_{\alpha} a_\alpha x^\alpha, \quad a_\alpha \in k,$$

som summerer over en endelig mængde af n-tupler $\alpha = (\alpha_1, \alpha_2, \ldots, \alpha_n)$. Mængden af alle polynomier i $x_1, x_2, \ldots, x_n$ med koefficienter i k skrives som $k[x_1, x_2, \ldots, x_n]$.

Følgende egenskaber gør sig gældende for polynomier.

Definition 2.3:
Lad $f = \sum_{\alpha} a_\alpha x^\alpha$ være et polynomium i $k[x_1, \ldots, x_n]$.

- *Vi kalder a_α koefficienten til monomiet x^α.*
- *Hvis $a_\alpha \neq 0$, så kalder vi $a_\alpha x^\alpha$ en term i f.*
- *Den totale grad af f, noteret $\deg(f)$, er den maksimale $|\alpha|$ hvor a_α er ikke-nul. Hvis f er nulpolynomiet så er $\deg(f) = -1$.*

Polynomier adderes ved for alle termer, der har samme monomium, at addere koefficienten. Multiplikation af polynomier foregår ved, at multiplicere hver term i det ene polynomium med alle termer i det andet polynomium. Et polynomium f siges, at dividere et andet polynomium g, hvis $g = fh$, hvor $h \in k[x_1, \ldots, x_n]$. Division af polynomier er dog mere kompliceret, hvorfor vi først må forstå hvordan polynomier skal ordnes, og dermed hvordan monomier relaterer til hinanden, for at kunne udfører division. Vi indfører derfor en ordning således, at det er klart hvilket monomium, der er størst.

Definition 2.4 (Monomial ordning):
En monomial ordning $>$ på $k[x_1, x_2, \ldots, x_n]$ er en relation på $\mathbb{Z}^n_{\geq 0}$, eller

ækvivalent en ordning på mængden af monomier $x^\alpha, \alpha \in \mathbb{Z}^n_{\geq 0}$ som opfylder følgende:

- $>$ *er en total ordning på $\mathbb{Z}^n_{\geq 0}$, hvilket betyder, at for alle elementer $\alpha, \beta \in \mathbb{Z}^n_{\geq 0}$ gælder, at enten er $\alpha > \beta$, $\beta > \alpha$ eller $\alpha = \beta$.*
- *Hvis $\alpha > \beta$ og $\gamma \in \mathbb{Z}^n_{\geq 0}$, så er $\alpha + \gamma > \beta + \gamma$*
- $>$ *er velordnet på $\mathbb{Z}^n_{\geq 0}$. Dette betyder at alle ikke-tomme delmængder af $\mathbb{Z}^n_{\geq 0}$ har et mindste element med hensyn til ordningen.*

Der findes mange forskellige ordninger. Vi vil her koncentrere os om tre af de mest anvendte.

Definition 2.5 (Leksikografisk ordning):
Lad $\alpha = (\alpha_1, \alpha_2, \ldots, \alpha_n)$ og $\beta = (\beta_1, \beta_2, \ldots, \beta_n) \in \mathbb{Z}^n_{\geq 0}$. Vi siger at $\alpha >_{lex} \beta$ hvis værdien længst til venstre i $\alpha - \beta \in \mathbb{Z}^n$ som ikke er nul er positiv. Vi skriver $x^\alpha >_{lex} x^\beta$ hvis $\alpha >_{lex} \beta$

Lad os betragte polynomiet $f = 8x^2yz^2 + 2x^3z - 5x^2y^2$ og forsøge at sortere monomierne efter leksikografisk ordning med $x > y > z$. De tre monomier i polynomiet kan skrives på følgende måde $x^{(2,1,2)}$, $x^{(3,0,1)}$ og $x^{(2,2,0)}$, hvilket betyder, at monomierne i polynomiet skal ordnes således $f = 2x^3z - 5x^2y^2 + 8x^2yz^2$.

Definition 2.6 (Gradueret leksikografisk ordning):
Lad $\alpha, \beta \in \mathbb{Z}^n_{\geq 0}$. Vi siger, at $\alpha >_{grlex} \beta$ hvis

$$|\alpha| = \sum_{i=1}^{n} \alpha_i > |\beta| = \sum_{i=1}^{n} \beta_i \quad \textit{eller} \quad |\alpha| = |\beta| \ \textit{ og } \alpha >_{lex} \beta$$

Lad os igen betragte polynomiet $f = 8x^2yz^2 + 2x^3z - 5x^2y^2$ og forsøge at sortere monomierne efter gradueret leksikografisk ordning med $x > y > z$.

For de tre monomier i polynomiet gælder, at $|(2,1,2)| = 5$, $|(3,0,1)| = 4$ og $|(2,2,0)| = 4$, hvilket betyder, at monomierne i polynomiet skal ordnes således $f = 8x^2yz^2 + 2x^3z - 5x^2y^2$.

Definition 2.7 (Gradueret omvendt leksikografisk ordning):
Lad $\alpha, \beta \in \mathbb{Z}^n_{\geq 0}$. Vi siger, at $\alpha >_{grevlex} \beta$ hvis

$$|\alpha| = \sum_{i=1}^{n} \alpha_i > |\beta| = \sum_{i=1}^{n} \beta_i \quad \textit{eller} \quad |\alpha| = |\beta| \ \textit{og}$$

værdien længst til højre i $\alpha - \beta \in \mathbb{Z}^n$ som ikke er nul er negativ.

Lad os for sidste gang forsøge, at ordne polynomiet $f = 8x^2yz^2 + 2x^3z - 5x^2y^2$, denne gang efter gradueret omvendt leksikografisk ordning med $x > y > z$. Af de foregående to eksempler ses hurtigt, at monomierne i polynomiet skal ordnes således $f = 8x^2yz^2 - 5x^2y^2 + 2x^3z$.

Definition 2.8:
Lad $f = \sum_{\alpha} a_\alpha x^\alpha$ være et ikke-nul polynomium i $k[x_1, x_2, \ldots, x_n]$ og lad $>$ være en monomial ordning.

- *Multigraden af f er*
$$\textit{multideg}(f) = max_>(\alpha \in \mathbb{Z}^n_{\geq 0} \mid a_\alpha \neq 0)$$
- *Den ledende koefficient af f er*
$$LC(f) = a_{multideg(f)} \in k$$
- *Det ledende monomium af f er*
$$LM(f) = x^{multideg(f)}$$
- *Den ledende term af f er*
$$LT(f) = LC(f) \cdot LM(f)$$

Hvis vi lader $f = 8x^2yz^2 + 2x^3z - 5x^2y^2$ og $>$ være leksikografisk ordning har vi, at

$$\begin{aligned} \text{multideg}(f) &= (3,0,1) \\ \text{LC}(f) &= 2 \\ \text{LM}(f) &= x^3z \\ \text{LT}(f) &= 2x^3z \end{aligned}$$

Vi er nu klar til at dividere to polynomier

Sætning 2.9 (Divisionsalgoritmen i flere variable):
Vælg en monomial ordning $>$ på $\mathbb{Z}^n_{\geq 0}$ og lad $F = (f_1, f_2, \ldots, f_s)$ være en ordnet s-tuple af polynomier i $k[x_1, x_2, \ldots, x_n]$. Så kan alle $f \in k[x_1, x_2, \ldots, x_n]$ skrives som

$$f = a_1f_1 + a_2f_2 + \cdots + a_sf_s + r,$$

hvor $a_i, r \in k[x_1, x_2, \ldots, x_n]$, og vi enten har, at $r = 0$ eller, at r er en linear kombination af monomier med koefficienter i k, hvor ingen af disse monomier kan divideres med nogle af $LT(f_1), LT(f_2), \ldots, LT(f_s)$. Vi kalder r resten af f ved division med F. Ydermere har vi, at hvis $a_if_i \neq 0$ så gælder

$$multideg(f) \geq multideg(a_if_i)$$

Beviset kan findes i [2].
Algoritme 2 viser hvordan division af to polynomier foregår.

2.2 Idealer og Gröbnerbaser

Lad os nu bevæge os videre frem mod definitionen af en Gröbnerbasis, og starte med et par indledende definitioner.

Definition 2.10 (Ideal):
En delmængde $I \subseteq k[x_1, x_2, \ldots, x_n]$ er et ideal hvis det opfylder

- $0 \in I$
- *Hvis $f, g \in I$ så er $f + g \in I$*

Algoritme 2: Divisionsalgoritme i $k[x_1, x_2, \ldots, x_n]$

Input: $f_1, \ldots, f_s, f$
Output: $a_1, \ldots, a_s, r$
$a_1 := 0; \ldots; a_s := 0; r := 0$
$p := f$
while $p \neq 0$ **do**
    $i := 1$
    divisionLavet := false
    while $i \leq s$ *og divisionLavet = false* **do**
        if *LT*(f_i) *deler LT*(p) **then**
            $a_i := a_i + \mathrm{LT}(p)/\mathrm{LT}(f_i)$
            $p := p - (\mathrm{LT}(p)/\mathrm{LT}(f_i))f_i$
            divisionLavet := true
        else
            $i := i + 1$
        end
    end
    if *divisionLavet = false* **then**
        $r := r + \mathrm{LT}(p)$
        $p := p - \mathrm{LT}(p)$
    end
end

- *Hvis $f \in I$ og $h \in k[x_1, x_2, \ldots, x_n]$ så er $hf \in I$*

Mængden, som genereres af en række polynomier, er givet ved følgende definition.

Definition 2.11:
Lad $f_1, f_2, \ldots, f_s$ være polynomier i $k[x_1, \ldots, x_n]$. Vi lader da

$$\langle f_1, \ldots, f_s \rangle = \left\{ \sum_{i=1}^{s} h_i f_i \mid h_1, \ldots, h_s \in k[x_1, \ldots, x_n] \right\}$$

Det er vigtigt at bemærke, at $\langle f_1, \ldots, f_s \rangle$ er et ideal.

Lemma 2.12:
Hvis $f_1, f_2, \ldots, f_s \in k[x_1, \ldots, x_n]$, så er $\langle f_1, \ldots, f_s \rangle$ et ideal i $k[x_1, \ldots, x_n]$. Vi siger, at $\langle f_1, \ldots, f_s \rangle$ er idealet, der genereres af $f_1, \ldots, f_s$.

Bevis:
Vi skal bevise, at $\langle f_1, \ldots, f_s \rangle$ opfylder de tre krav for et ideal.

For det første er $0 \in \langle f_1, \ldots, f_s \rangle$, idet $0 = \sum_{i=1}^{s} 0 \cdot f_i$.

Antag nu, at $g = \sum_{i=1}^{s} p_i \cdot f_i$ og $j = \sum_{i=1}^{s} q_i \cdot f_i$ og lad $h \in k[x_1, \ldots, x_n]$. Så udgør ligningerne

$$g + j = \sum_{i=1}^{s} (p_i + q_i) f_i,$$

og

$$hg = \sum_{i=1}^{s} (hp_i) f_i$$

resten af beviset.

■

Lad os udvide vores viden om idealer til også at omfatte monomial idealer og deres egenskaber.

Definition 2.13 (Monomial ideal):
Et ideal $I \subseteq k[x_1, \ldots, x_n]$ er et monomial ideal hvis der er en delmængde $A \subset \mathbb{Z}^n_{\geq 0}$ således, at I består af alle polynomier, som er endelige summer på formen $\sum_{a \in A} h_\alpha x^\alpha$ hvor $h_\alpha \in k[x_1, \ldots, x_n]$. Hvis dette er tilfældet skrives det som $I = \langle x^\alpha \mid \alpha \in A \rangle$.

Vi ser at notationen i definition 2.11 og 2.13 ligner hinanden. Forskellen på disse definitioner er, at i 2.13 er det et krav, at det er monomier fremfor polynomier og det er tilladt, at der er uendelig mange af dem.

Lemma 2.14:
Lad $I = \langle x^\alpha \mid \alpha \in A \rangle$ være et monomial ideal. Så er et monomium x^β i I hvis og kun hvis x^β kan deles af et x^α hvor $\alpha \in A$.

Bevis:
Hvis der findes et $\alpha \in A$ således, at x^α deler x^β betyder det, at der findes et $h \in k[x_1, \ldots, x_n]$ således at $x^\beta = hx^\alpha$ og dermed er $x^\beta \in I$.
Hvis $x^\beta \in I$ så er $x^\beta = \sum_{i=1}^{s} h_i x^{\alpha_i}$, hvor $h_i \in k[x_1, \ldots x_n]$ og $\alpha_i \in A$. Vi kan nu se at der for hvert led på højresiden findes et x^{α_i}, som deler leddet, og derfor må det også gælde for venstresiden.

■

Vi har følgende definition omkring de ledende elementer i idealer.

Definition 2.15:
Lad $I \subseteq k[x_1, \ldots, x_n]$ være et ideal (forskelligt fra $\{0\}$)

- *$LT(I)$ er mængden af ledende termer af elementer i I.*

$$LT(I) = \{cx^\alpha \mid \text{ der eksisterer } f \in I \text{ med } LT(f) = cx^\alpha\}$$

- *$\langle LT(I)\rangle$ er idealet genereret af elementer fra $LT(I)$.*

Der er nogle vigtige ting at fremhæve med hensyn til $\langle \mathrm{LT}(I)\rangle$. For eksempel hvis $I = \langle f_1, \ldots, f_s\rangle$ så gælder det, at $\langle \mathrm{LT}(f_1), \ldots, \mathrm{LT}(f_s)\rangle$ og $\langle \mathrm{LT}(I)\rangle$ ikke nødvendigvis er det samme ideal. Følgende sætning giver os vigtige egenskaber for $\langle \mathrm{LT}(I)\rangle$, beviset findes i [2].

Sætning 2.16:
Lad $I \subseteq k[x_1, \ldots, x_n]$ være et ideal.

1. *$\langle LT(I)\rangle$ er et monomial ideal.*
2. *Der findes $g_1, \ldots, g_t \in I$ således at $\langle LT(I)\rangle = \langle LT(g_1), \ldots, LT(g_t)\rangle$.*

Vi har følgende sætning som siger at alle idealer kan genereres af en endelig mængde polynomier.

Sætning 2.17 (Hilberts basissætning):
Ethvert ideal $I \subseteq k[x_1, \ldots, x_n]$ har en endelig genererende mængde, således at $I = \langle g_1, \ldots, g_t\rangle$ hvor $g_1, \ldots, g_t \in I$.

Bevis:
Hvis $I = \{0\}$, kan vi sætte den genererende mængde til at være $\{0\}$ som er endelig. Hvis I indeholder et ikke-nul polynomium ved vi fra sætning 2.16, at der findes $g_1, \ldots, g_t \in I$ således at $\langle \mathrm{LT}(I)\rangle = \langle \mathrm{LT}(g_1), \ldots, \mathrm{LT}(g_t)\rangle$, vi vil nu vise at $\langle g_1, \ldots, g_t\rangle = I$.
Det er tydeligt at $\langle g_1, \ldots, g_t\rangle \subset I$ da alle $g_i \in I$.
Vi tager nu et hvilket som helst polynomium $f \in I$. Ved at anvende divisionsalgoritmen, som er algoritme 2 til at dividere f med $\langle g_1, \ldots, g_t\rangle$ får vi et udtryk på formen

$$f = a_1 g_1 + \cdots + a_t g_t + r$$

hvor r ikke kan deles af $\mathrm{LT}(g_1), \ldots, \mathrm{LT}(g_t)$. Vi ser nu at

$$r = f - a_1 g_1 - \cdots - a_t g_t$$

Så hvis $r \neq 0$ må $\mathrm{LT}(r) \in \langle \mathrm{LT}(I) \rangle = \langle \mathrm{LT}(g_1), \ldots, \mathrm{LT}(g_t) \rangle$ og fra lemma 2.14 at $\mathrm{LT}(r)$ må være delelig med et $\mathrm{LT}(g_i)$. Dette er i modstrid med at r er resten ved division og derfor må $r = 0$. Vi har dermed

$$f = a_1 g_1 + \cdots + a_t g_t + 0 \in \langle g_1, \ldots g_t \rangle$$

hvilket viser at $I \subset \langle g_1, \ldots g_t \rangle$ hvilket beviser sætningen.

■

Vi kan nu vise følgende sætning.

Sætning 2.18:
Lad

$$I_1 \subseteq I_2 \subseteq I_3 \subseteq \cdots$$

være en voksende kæde af idealer i $k[x_1, \ldots, x_n]$. *Så eksisterer der et* $N \geq 1$ *således, at*

$$I_N = I_{N+1} = I_{N+2} = \cdots$$

Bevis:
Givet den voksende kæde af idealer $I_1 \subseteq I_2 \subseteq I_3 \subseteq \cdots$, betragt nu mængden $I = \bigcup_{i=1}^{\infty} I_i$. Vi vil starte med at vise, at I er et ideal. Vi ser først, at $0 \in I$ da $0 \in I_i$ for alle i, efterfølgende ser vi, at hvis $f, g \in I$ må det gælde, at $f \in I_i$ og $g \in I_j$. Hvis $i = j$ har vi $g + f \in I_j$ og dermed også i I. Hvis $i \neq j$, kan vi da I_i og I_j er fra den voksende kæde, uden tab af generalitet antage at $i < j$ så $f, g \in I_j$ og dermed er $g + f \in I_j$ og dermed også i I. Vi kan ligeledes vise, at hvis $f \in I$ og $r \in k[x_1, \ldots, x_n]$ så er $f \cdot r \in I$. Som før må det gælde, at $f \in I_i$ hvilket medfører, at $f \cdot r \in I_i \subseteq I$. Vi har hermed vist, at I er et ideal.
Vi ved fra sætning 2.17, at idealet I har en endelig genererende mængde så $I = \langle f_1, \ldots, f_s \rangle$. Der må gælde, at ethvert af de genererende elementer ligger i et I_j, så $f_i \in I_{j_i}$ for nogle $j_i, i = 1, \ldots, s$. Vi sætter nu N til at være det største j_i. Så har vi på grund af den voksende kæde at $f_i \in I_N$ for alle i. Vi har derfor

$$I = \langle f_1, \ldots f_s \rangle \subseteq I_N \subseteq I_{N+1} \subseteq \cdots \subseteq I$$

Vi kan altså se at den voksende kæde stabiliserer ved idealet I_N og alle de efterfølgende idealer i kæden er ens.

■

Vi er nu klar til at definere en Gröbnerbasis.

Definition 2.19 (Gröbnerbasis):
Vælg en monomial ordning. En endelig delmængde $G = \{g_1, \ldots, g_t\}$ af et ideal I kaldes en Gröbnerbasis hvis

$$\langle LT(g_1), \ldots, LT(g_t)\rangle = \langle LT(I)\rangle$$

Vi ved fra sætning 2.16, at der findes en Gröbnerbasis for alle idealer.
Vi vil nu vise nogle egenskaber ved Gröbnerbaser, og at ordningen ved division med en Gröbnerbasis er ligegyldig.

Sætning 2.20:
Lad $G = \{g_1, \ldots, g_t\}$ være en Gröbnerbasis for et ideal $I \subseteq k[x_1, \ldots, x_n]$ og $f \in k[x_1, \ldots, x_n]$. Så eksisterer der et unikt $r \in k[x_1, \ldots, x_n]$ med følgende egenskaber

i) Ingen af termerne i r kan deles af $LT(g_1), \ldots, LT(g_t)$

ii) Der eksisterer et $g \in I$ sådan at $f = g + r$.

Særligt gælder der at r er resten når f divideres med G ved anvendelse af divisionsalgoritmen, uanset hvordan elementerne i G er ordnet.

Bevis:
Divisionsalgoritmen giver $f = a_1g_1 + \cdots + a_tg_t + r$ hvor r opfylder i), og vi kan opfylde ii) ved at sætte $g = a_1g_1 + \cdots + a_tg_t \in I$. Vi har nu vist at et r eksisterer og mangler nu at vise at det er unikt.
For at vise at r er unik, antager vi at $f = g + r = g' + r'$, hvor r og r' opfylder i) og ii). Så må der gælde $r - r' = g' - g \in I$ og hvis $r \neq r'$ har vi $\mathrm{LT}(r - r') \in \langle \mathrm{LT}(I)\rangle = \langle \mathrm{LT}(g_1), \ldots, \mathrm{LT}(g_t)\rangle$. Fra lemma 2.14 følger det at $\mathrm{LT}(r - r')$ kan deles af et $\mathrm{LT}(g_i)$. Dette kan ikke lade sig gøre da der ikke er en term i r eller r' som kan deles af et $\mathrm{LT}(g_i)$, derfor må $r - r' = 0$ og dermed er r unik.
Resten følger af at r er unik.

■

Korollar 2.21:
Lad $G = \{g_1, \ldots, g_t\}$ være en Gröbnerbasis for idealet $I \subseteq k[x_1, \ldots, x_n]$ og lad

$f \in k[x_1, \ldots, x_n]$, så er $f \in I$ hvis og kun hvis resten ved division af f med G er nul.

Bevis:
Hvis resten er nul er f på formen $f = a_1 g_1 + \cdots + a_t g_t$ og da $a_i g_i \in I$ følger der at $f \in I$.
Hvis der omvendt gælder at $f \in I$ så opfylder $f = f + 0$ de to punkter i sætning 2.20. Hvilket betyder at 0 er den unikke rest af f ved division med G.

■

Lad os indføre følgende om division med mængder.

Definition 2.22:
Vi skriver $\overline{f}^F$ som resten ved division af f med den ordnede s-tuple $F = (f_1, f_2, \ldots, f_s)$.

Hvis F er en Gröbnerbasis for $\langle f_1, f_2, \ldots, f_s \rangle$ ved vi fra sætning 2.20 at F kan opfattes som en mængde uden en bestemt ordning.
Vi vil nu undersøge nogle egenskaber ved $\overline{f}^F$.

Sætning 2.23:
Lad $I \subseteq k[x_1, \ldots, x_n]$ være et ideal, G en Gröbnerbasis for I og $f, g \in k[x_1, \ldots, x_n]$ og $c \in k$. Så er $\overline{f+g}^G = \overline{f}^G + \overline{g}^G$ og $\overline{c \cdot f}^G = c \cdot \overline{f}^G$.

Bevis:
Vi ved fra 2.20, at der findes unikke r_1 og r_2 således, at

$$f = h_1 + r_1$$
$$g = h_2 + r_2$$

dermed er

$$f + g = (h_1 + h_2) + (r_1 + r_2) = h_3 + r_3,$$

hvor $h_3 \in I$ og $r_3 \notin \langle \mathrm{LT}(I) \rangle$. Derfor får vi følgende når vi deler med Gröbner-

basen

$$\overline{f}^G = r_1$$
$$\overline{g}^G = r_2$$
$$\overline{f+g}^G = r_3 = r_1 + r_2$$

Vi ser nu at $\overline{f}^G + \overline{g}^G = \overline{f+g}^G$.
På samme måde har vi

$$c \cdot f = c \cdot h_1 + c \cdot r_1,$$

hvor $c \cdot h_1 \in I$ og $c \cdot r_1 \notin \langle \mathrm{LT}(I) \rangle$. Derfor får vi følgende når vi deler med Gröbnerbasen

$$\overline{f}^G = r_1$$
$$\overline{c \cdot f}^G = c \cdot r_1$$

Vi har således at

$$c \cdot \overline{f}^G = \overline{c \cdot f}^G$$

.

■

Vi er interesserede i at kunne finde en Gröbnerbasis for et ideal, men før vi kan det skal vi først have følgende definition.

Definition 2.24:
Lad $f, g \in k[x_1, \ldots, x_n]$ være ikke-nul polynomier.

- *Hvis multideg$(f) = \alpha$ og multideg$(g) = \beta$, så lad $\gamma = (\gamma_1, \ldots, \gamma_n)$, hvor $\gamma_i = \max(\alpha_i, \beta_i)$ for alle i. Vi siger, at x^γ er det mindste fælles multiplum af LM(f) og LM(g). Dette skrives som*

  $$x^\gamma = Lcm(LM(f), LM(g))$$

  .

- *S-polynomiet af f og g er defineret som følgende*

  $$S(f,g) = \frac{x^\gamma}{LT(f)} \cdot f - \frac{x^\gamma}{LT(g)} \cdot g.$$

Lad os nu se på sammenhængen mellem en Gröbnerbasis og S-polynomiet. Følgende sætning er bevist i [2].

Sætning 2.25:
Lad I være et polynomial ideal. Så er en basis $G = \{g_1, \ldots, g_t\}$ for I en Gröbnerbasis hvis og kun hvis der for alle par $i \neq j$ gælder at resten efter division af $S(g_i, g_j)$ med G er nul.

Vi vil i det følgende bruge denne notation. Hvis $G = \{g_1, \ldots, g_t\}$ så vil $\langle G \rangle$ og $\langle \mathrm{LT}(G) \rangle$ noterer følgende idealer

$$\langle G \rangle = \langle g_1, \ldots, g_t \rangle$$
$$\langle \mathrm{LT}(G) \rangle = \langle \mathrm{LT}(g_1), \ldots, \mathrm{LT}(g_t) \rangle$$

Lad os nu betragte hvordan vi kan konstruere en Gröbnerbasis.

Sætning 2.26:
Lad $I = \langle f_1, \ldots, f_s \rangle \neq 0$ være et polynomial ideal, så kan en Gröbnerbasis konstrueres i et endeligt antal skridt ved brug af Buchbergers algoritme, som er algoritme 3

Algoritme 3: Buchbergers algoritme

Input: $F = (f_1, \ldots, f_s)$
Output: En Gröbnerbasis $G = (g_1, \ldots, g_t)$ for I
$G = F$
repeat
  $G' = G$
  for *Alle par* $\{p, q\}, p \neq q$ *i* G' **do**
    $S = \overline{S(p,q)}^{G'}$
    if $S \neq 0$ **then**
      $G = G \cup \{S\}$
    end
  end
until $G = G'$

Bevis:
Vi vil starte med at vise, at $G \in I$ holder i alle skridt af algoritmen. Dette

gør sig gældende fra starten, og for hver gang vi udvider G gør vi det med $S = \overline{S(p,q)}^{G'}$ for $p, q \in G$. Så hvis $G \subset I$ må p, q og derfor $S(p,q)$ også være i I, og siden vi dividere med $G' \subset I$ får vi at $G \cup \{S\} \subset I$. Da G stadig indeholder den givne basis F for I, ser vi at G er en basis for I. Algoritmen stopper når $G = G'$, hvilket betyder at $S = \overline{S(p,q)}^{G'} = 0$ for alle $p, q \in G$. Vi ved nu fra sætning 2.25 at G er en Gröbnerbasis for idealet $\langle G \rangle = I$.
Vi mangler nu blot at vise, at vi kan være sikre på, at algoritmen stopper. Vi er nødt til at undersøge hvad der sker hver gang algoritmen kører gennem loopet. Da vi til enhver tid har $G' \subseteq G$, så må

$$\langle \mathrm{LT}(G') \rangle \subseteq \langle \mathrm{LT}(G) \rangle. \tag{2.1}$$

Hvis $G' \neq G$ påstår vi, at $\langle \mathrm{LT}(G') \rangle$ er strengt mindre end $\langle \mathrm{LT}(G) \rangle$. For at vise, at denne påstand er korrekt antag ,at en ikke-nul rest af S-polynomiet er blevet lagt til G. Da r er resten ved division med G' ved vi at $\mathrm{LT}(r) \notin \langle \mathrm{LT}(G') \rangle$. Samtidig har vi at $\mathrm{LT}(r) \in \langle \mathrm{LT}(G) \rangle$ hvilket beviser vores påstand.
Da vi hver gang vi kører gennem loopet sætter $G' = G$ får vi fra ligning 2.1, at $\langle \mathrm{LT}(G') \rangle$ danner en voksende kæde af idealer i $k[x_1, \ldots, x_n]$. Vi har således fra sætning 2.18, at efter et endeligt antal iterationer vil kæden stabiliserer således, at vi er sikre på, at $\langle \mathrm{LT}(G') \rangle = \langle \mathrm{LT}(G) \rangle$ vil ske. Hvilket betyder at der ikke længere er en rest r som kan bliver tilføjet som medfører at $G = G'$, vi er nu sikre på at algoritmen stopper efter et endeligt antal iterationer.

■

2.3 Kongruens og kvotient

Vi vil nu definere kongruens mellem to polynomier.

Definition 2.27:
Lad $I \subseteq k[x_1, \ldots, x_n]$ være et ideal og $f, g \in k[x_1, \ldots, x_n]$ så siger vi, at f og g er kongruente modulo I hvis $f - g \in I$. Vi noterer dette

$$f \equiv g \pmod{I}$$

Vi vil nu se på et eksempel med to kongruente polynomier.

Eksempel 2.28:
Hvis $I = \langle x^3 + y^2 + x, x^2 + xy \rangle \subset k[x, y]$ så er $f = x^2y + xy^4 - y^3 - y^2 - x$ og $g = -x^2y^3 + x^3y + xy$ kongruente modulo I da

$$\begin{aligned} f - g &= x^2y + xy^4 - y^3 - y^2 - x + x^2y^3 - x^3y - xy \\ &= x^2y + xy^4 - y^3 - y^2 - x + x^3 - x^3 + x^2y^3 - x^3y - xy \\ &= (x^3 + x^2y + x^2y^3 + xy^4) - (x^3y + y^3 + xy + x^3 + y^2 + x) \\ &= (x + y^3)(x^2 + xy) - (y + 1)(x^3 + y^2 + x) \in I \end{aligned}$$

Vi vil i følgende sætning vise en vigtig egenskab for kongruensrelationen.

Sætning 2.29:
Lad $I \subseteq k[x_1, \ldots, x_n]$ være et ideal, så er kongruens modulo I en ækvivalensrelation.

Bevis:
For at vise, at kongruens modulo I er en ækvivalensrelation, skal vi vise, at den er refleksiv, symmetrisk og transitiv. Vi starter med at vise, at den er refleksiv, altså $f \equiv f \pmod{I}$.
Da $f - f = 0 \in I$ er kongruens modulo I refleksiv. For at vise symmetri antager vi at $f \equiv g \pmod{I}$ hvilket betyder at $f - g \in I$ og dermed $g - f = (-1)(f - g) \in I$. Vi mangler nu kun at vise, at kongruens modulo I er transitiv, antag derfor at $f \equiv g \pmod{I}$ og $g \equiv h \pmod{I}$ så har vi at $f - g, g - h \in I$ og da I er lukket under addition må der gælde at $f - h = f - g + g - h \in I$ og dermed er $f \equiv h \pmod{I}$.

■

En ækvivalensrelation på en mængde S, deler S op i disjunkte delmængder som kaldes ækvivalensklasser. For ethvert $f \in k[x_1, \ldots, x_n]$ har vi, at den klasse f tilhører er mængden

$$[f] = \{g \in k[x_1, \ldots, x_n] \mid g \equiv f \pmod{I}\}$$

Vi kan nu definere kvotienten.

Definition 2.30:
Kvotienten af $k[x_1, \ldots, x_n]$ modulo I skrevet som $k[x_1, \ldots, x_n]/I$, er mængden af ækvivalensklasser af kongruens modulo I

$$k[x_1, \ldots, x_n]/I = \{[f] \mid f \in k[x_1, \ldots, x_n]\}$$

Antag vi har $k = \mathbb{R}, n = 1$ og $I = \langle x^2 - 2 \rangle$. Vi ønsker nu at beskrive alle ækvivalensklasser af kongruens modulo I. Vi ved fra divisionsalgoritmen at alle $f \in \mathbb{R}[x]$ kan skrives som $f = q \cdot (x^2 - 2) + r$ hvor $r = ax + b$ for $a, b \in \mathbb{R}$. Da $f - r = q \cdot (x^2 - 2) \in I$ har vi at $f \equiv r \pmod{I}$. Dermed gælder det, at alle elementer i $\mathbb{R}[x]$ tilhører en ækvivalensklasse på formen $[ax + b]$ og $\mathbb{R}[x]/I = \{[ax + b] \mid a, b \in \mathbb{R}\}$.
Idet $k[x_1, \ldots, x_n]$ er en ring, kan vi, givet to klasser $[f], [g] \in k[x_1, \ldots, x_n]/I$, forsøge at definere addition og multiplikation for klasser ved hjælp af de tilsvarende operationer på elementer i $k[x_1, \ldots, x_n]$

$$[f] + [g] = [f + g] \tag{2.2}$$
$$[f] \cdot [g] = [f \cdot g] \tag{2.3}$$

Vi skal nu tjekke om disse formler giver mening. Vi er derfor nødt til at vise, at hvis vi vælger andre $f' \in [f]$ og $g' \in [g]$ så er klassen $[f' + g']$ den samme som $[f + g]$. Ligeledes skal vi tjekke at $[f' \cdot g'] = [f \cdot g]$.

Sætning 2.31:
Operationerne defineret i ligningerne (2.2) *og* (2.3) *giver samme klasse i $k[x_1, \ldots, x_n]/I$ på højresiden lige meget hvilket $f' \in [f]$ og $g' \in [g]$ vi bruger. Vi siger, at operationerne i* (2.2) *og* (2.3) *er veldefinerede.*

Bevis:
Hvis $f' \in [f]$ og $g' \in [g]$, så er $f' = f + a$ og $g' = g + b$ hvor $a, b \in I$. Dette giver

$$f' + g' = (f + a) + (g + b) = (f + g) + (a + b).$$

Da $a + b \in I$ følger det, at $f' + g' \equiv f + g \pmod{I}$ hvilket betyder at $[f' + g'] = [f + g]$. På samme måde har vi

$$f' \cdot g' = (f + a) \cdot (g + b) = fg + fb + ag + ab.$$

Da $a, b \in I$ har vi at $ag + fb + ab \in I$ af dette følger at $f' \cdot g' \equiv f \cdot g \pmod{I}$ hvilket betyder at $[f' \cdot g'] = [f \cdot g]$.

■

Følgende sætning viser sig, at kvotienten er en kommutativ ring. Beviset findes i [2].

Sætning 2.32:
Lad $I \subseteq k[x_1, \ldots, x_n]$ være et ideal. Kvotienten $k[x_1, \ldots, x_n]/I$ er en kommutativ ring med addition og multiplikation defineret som i ligningerne (2.2) *og* (2.3)*.*

Lad os derfor definere følgende omkring kommutative ringe.

Definition 2.33:
Lad R, S være kommutative ringe.

1. *En afbildning $\phi : R \to S$ er en ringisomorfi hvis*
   (a) *ϕ bevarer addition: $\phi(r + r') = \phi(r) + \phi(r')$ for alle $r, r' \in R$*
   (b) *ϕ bevarer multiplikation: $\phi(r \cdot r') = \phi(r) \cdot \phi(r')$ for alle $r, r' \in R$*
   (c) *ϕ er bijektiv (injektiv og surjektiv)*
2. *To ringe er isomorfe hvis der eksisterer en isomorfi $\phi : R \to S$. Vi skriver $R \cong S$ for at beskrive, at R er isomorf med S.*
3. *En afbildning $\phi : R \to S$ er en ringhomomorfi hvis ϕ har egenskab 1a og 1b, men ikke nødvendigvis 1c, og ϕ yderligere afbilleder den multiplikative invers $1 \in R$ til $1 \in S$.*

Vi har følgende for idealet af ledende termer.

Sætning 2.34:
Fastsæt en monomial ordning på $k[x_1, \ldots, x_n]$ og lad $I \subseteq k[x_1, \ldots, x_n]$ være et ideal. Lad $\langle LT(I) \rangle$ være idealet genereret af de ledende termer i I. Så gælder det, at

i) ethvert $f \in k[x_1, \ldots, x_n]$ er kongruent modulo I til et unikt polynomium r som er en linearkombination af monomier i komplimentet til $\langle LT(I) \rangle$.

ii) elementerne fra $\{x^\alpha \mid x^\alpha \notin \langle LT(I) \rangle\}$ er "lineært uafhængige modulo I". Det vil sige, at hvis det gælder, at

$$\sum_\alpha c_\alpha x^\alpha \equiv 0 \pmod{I},$$

hvor x^α ligger i komplementet af $\langle LT(I) \rangle$, så er $c_\alpha = 0$ for alle α.

Bevis:
Ad *i*) Lad G være en Gröbnerbasis for I og $f \in k[x_1, \ldots, x_n]$. Ved at bruge divisionsalgoritmen får vi resten $r = \overline{f}^G$ som opfylder $f = q + r$ hvor $q \in I$. Dette medfører at $f - r = q \in I$ så $f \equiv r \pmod{I}$. Divisionsalgoritmen fortæller os at r er en linearkombination af monomier $x^\alpha \notin \langle \mathrm{LT}(I) \rangle$. At r er unik kommer fra sætning 2.20.
Ad *ii*) Lad $H(x) = \sum_\alpha c_\alpha x^\alpha$. Antag, at $H(x) \equiv 0 \pmod{I}$, vi ved at $H'(x) = \sum_\alpha 0x^\alpha \equiv 0 \pmod{I}$, vi har nu således to polynomier begge er ækvivalente med 0 modulo I hvilket er i modstrid med i) og dermed må $H(x) = H'(x)$ og alle $c_\alpha = 0$ i $H(x)$.

■

Følgende gælder om et ideal.

Sætning 2.35:
Lad $I \subseteq k[x_1, \ldots, x_n]$ være et ideal, så er $k[x_1, \ldots, x_n]/I$ isomorf med $S = span(x^\alpha \mid x^\alpha \notin \langle LT(I) \rangle)$ som et vektorrum over $\mathbb{F}_q$.

Bevis:
Vi ved fra sætning 2.34, at afbildningen $\phi : k[x_1, \ldots, x_n]/I \to S$ er bijektiv mellem klasserne i $k[x_1, \ldots, x_n]/I$ og elementerne i S.
Vi mangler nu at vise at ϕ bevarer addition og multiplikation. Betragt addition defineret som i 1a fra definition 2.33. Hvis $[f], [g]$ er elementer i $k[x_1, \ldots, x_n]/I$ så kan vi ved at bruge sætning 2.34 "standardisere" den polynomiale repræsentant ved at regne resten ved division med en Gröbnerbasis G for I. Vi ved fra 2.23 at $\overline{f+g}^G = \overline{f}^G + \overline{g}^G$ således, at hvis

$$\overline{f}^G = \sum_\alpha c_\alpha x^\alpha \quad \text{og} \quad \overline{g}^G = \sum_\alpha d_\alpha x^\alpha,$$

hvor der summeres over de α hvor $x^\alpha \notin \langle \mathrm{LT}(I) \rangle$. Så er

$$\overline{f+g}^G = \sum_\alpha (c_\alpha + d_\alpha) x^\alpha$$

Vi kan nu se, at med standard repræsentanterne er sumoperationen i $k[x_1, \ldots, x_n]$ den samme som vektorsummen i k-vektorrummet $S = \mathrm{span}(x^\alpha \mid x^\alpha \notin \langle \mathrm{LT}(I) \rangle)$.
Vi ved fra sætning 2.23 at hvis $c \in k$ så er $\overline{c \cdot f}^G = c \cdot \overline{f}^G$, af dette følger

$$\overline{c \cdot f}^G = \sum_\alpha c \cdot c_\alpha x^\alpha$$

dette viser at multiplikation med c i $k[x_1, \ldots, x_n]/I$ er som skalarmultiplikation i S. Vi har nu vist at afbildningen ϕ er en vektorrumsisomorfi.

■

2.4 Varietet

Vi vil nu definere varietet.

Definition 2.36 (Varietet):
Lad k være et legeme og lad $f_1, \ldots, f_s$ være polynomier i $k[x_1, \ldots, x_n]$. Så sætter vi

$$V(f_1, \ldots, f_s) = \{(a_1, \ldots, a_n) \in k^n \mid f_i(a_1, \ldots, a_n) = 0 \text{ for alle } 1 \leq i \leq s\}.$$

Vi kalder $V(f_1, \ldots, f_s)$ for varieteten af $f_1, \ldots, f_s$.

Varieteten af $f_1, \ldots, f_s$ er således mængden af fælles nulpunkter for polynomierne $f_1, \ldots, f_s$. Vi vil nu kigge på alle de polynomier der evaluere til nul over en varietet.

Definition 2.37:
Lad $V \subseteq k^n$ være en varietet, så er mængden

$$I(V) = \{f \in k[x_1, \ldots, x_n] \mid f(a_1, \ldots a_n) = 0 \text{ for alle } (a_1, \ldots, a_n) \in V\}$$

Vi ønsker nu at vise, at $I(V)$ er et ideal.

Sætning 2.38:
Hvis $V \subseteq k^n$ er en varietet, så er $I(V) \subseteq k[x_1, \ldots, x_n]$ et ideal. Vi vil kalde $I(V)$ for idealet af V.

Bevis:
For at vise, at $I(V)$ er et ideal skal vi vise, at $0 \in I(V)$, $f + g \in I(V)$ og at $hf \in I(V)$, for alle $f, g \in I(V)$ og $h \in k[x_1, \ldots, x_n]$.
Vi ser, at $0 \in I(V)$ da nulpolynomiet evaluerer til nul over hele $k[x_1, \ldots, x_n]$ og dermed også over $I(V)$. Vi antager nu at $f, g \in I(V)$ og $h \in k[x_1, \ldots, x_n]$. Lad $(a_1, \ldots, a_n)$ være et tilfældigt punkt i V, så har vi at

$$\begin{aligned} f(a_1, \ldots, a_n) + g(a_1, \ldots, a_n) &= 0 + 0 = 0 \\ h(a_1, \ldots, a_n) f(a_1, \ldots, a_n) &= h(a_1, \ldots, a_n) \cdot 0 = 0 \end{aligned}$$

og dermed er $I(V)$ et ideal.

■

Vi indfører følgende sætning omkring sammenhængen mellem varieteten af et ideal og idealets størrelse, beviset findes i [2].

Sætning 2.39 (Hilberts svage Nullstellensatz):
Lad k være et algebraisk aflukket legeme og lad $I \subseteq k[x_1, \ldots, x_n]$ være et ideal så er $V(I) = \emptyset$ hvis og kun hvis $I = k[x_1, \ldots, x_n]$.

Vi definerer fodaftrykket.

Definition 2.40 (Fodaftryk):
Lad $>$ være en monomial ordning og $I \subseteq k[x_1, \ldots, x_n]$ et ideal. Fodaftrykket af I med hensyn til $>$ er mængden af monomier, der ikke kan findes som ledende monomier af nogle polynomier i idealet. Fodaftrykket noteres $\Delta_>(I)$.

Vi har altså, at

$$\Delta_>(I) = \{x^\alpha \mid x^\alpha \notin \langle \mathrm{LT}(I) \rangle\}$$

Vi vil nu se på et eksempel på et fodaftryk.

Eksempel 2.41:
Hvis vi har et ideal $I \in k[x,y]$ med Gröbnerbasen $\langle x^4, y^3, x^2y\rangle$ og $>_{\text{lex}}$ som den monomielle ordning. Så er fodaftrykket

$$\Delta_{>_{\text{lex}}}(I) = \{1, x, x^2, x^3, xy, y, y^2, xy^2\}$$

Vi vil skrive $\bar{k}$ for den algebraiske aflukning af legemet k. Hvilket betyder, at $k[x] \subseteq \bar{k}[x]$ og alle rødder til polynomier i $k[x]$ ligger i $\bar{k}$. Vi vil nu definere Lagrangeinterpolation.

Definition 2.42 (Lagrangeinterpolation):
Vi ønsker at finde et polynomium F som opfylder, at $F(\alpha_i, \beta_i) = \gamma_i$ for $i = 1, \ldots, l$. Følgende polynomium opfylder dette.

$$F(x,y) = \sum_{i=1}^{l} \left(\frac{\prod_{\alpha_s \neq \alpha_i}(x - \alpha_s) \ \prod_{\beta_t \neq \beta_i}(y - \beta_t)}{\prod_{\alpha_s \neq \alpha_i}(\alpha_i - \alpha_s) \ \prod_{\beta_t \neq \beta_i}(\beta_i - \beta_t)} \cdot \gamma_i \right).$$

Det er ligetil at udvide dette til flere variabler.

Vi vil nu se på et eksempel med Lagrangeinterpolation.

Eksempel 2.43:
Vi ønsker at finde et polynomium F der opfylder

$$F(1,2) = 5 \quad F(2,4) = 6 \quad F(1,3) = 2$$

Vi får nu

$$\begin{aligned} F(x,y) = &\frac{(x-2)(y-3)(y-4)}{(1-2)(2-3)(2-4)} \cdot 5 + \frac{(x-1)(y-2)(y-3)}{(2-1)(4-2)(4-3)} \cdot 6 \\ &+ \frac{(x-2)(y-2)(y-4)}{(1-2)(3-2)(3-4)} \cdot 2 \end{aligned}$$

Vi ser nu, at hvis vi for eksempel regner $F(1,2)$ så får vi, at

$$\begin{aligned} F(1,2) &= \frac{(1-2)(2-3)(2-4)}{(1-2)(2-3)(2-4)} \cdot 5 + \frac{(1-1)(2-2)(2-3)}{(2-1)(4-2)(4-3)} \cdot 6 \\ &\quad + \frac{(1-2)(2-2)(2-4)}{(1-2)(3-2)(3-4)} \cdot 2 \\ &= 1 \cdot 5 + 0 \cdot 6 + 0 \cdot 2 \\ &= 5, \end{aligned}$$

hvilket er det ønskede resultat.

🦋

Med dette kan vi vise, at varieteten er stærkt relateret til fodaftrykket.

Sætning 2.44:
Hvis $I \subseteq k[x_1, \ldots, x_n]$ er et ideal hvor $\Delta_>(I)$ er endelig, så har varieteten $V_{\overline{k}}(I)$ (altså mængden af fælles nulpunkter i $\overline{k}^n$ af polynomierne i I) størrelse højest $|\Delta_>(I)|$.

Bevis:
Lad $\{P_1, \ldots, P_m\} \subseteq V_{\overline{k}}(I)$ og betragt afbildningen $\phi : \overline{k}[x_1, \ldots, x_n]/I \to \overline{k}^m$ givet ved $\phi(F) = (F(P_1), \ldots, F(P_m))$.
Vi ser at funktionen ϕ er surjektiv da det altid er muligt, ved hjælp af Lagrangeinterpolation, at finde et polynomium F, der evaluerer til en vilkårlig m-tuple $(\gamma_1, \ldots, \gamma_m)$ uanset hvilke P_i'er der er valgt. Vi må have, at $\dim(\overline{k}[x_1, \ldots, x_n]/I) \geq \dim(\overline{k}^m) = m$, da F er surjektiv.
Vi kan uden tab af generalitet antage at $F \subseteq \mathit{Span}_{\overline{k}}(\Delta_>(I))$. Vi har ifølge sætning 2.35, at $\dim(\overline{k}[x_1, \ldots, x_n]/I) = \dim(\mathit{Span}(\Delta_>(I)))$, og størrelsen af $\Delta_>(I)$ må derfor være større end m. Da vi kan vælge $\{P_1, \ldots, P_m\} = V_{\overline{k}}(I)$ må det gælde, at varieteten ikke kan være større end fodaftrykket.

■

Korollar 2.45:
Lad $f(x_1, \ldots, x_n) \in k[x_1, \ldots, x_n]$ være et ikke-nulpolynomium og fastsæt en monomial ordning $>$. Her er k et legeme som indeholder $\mathbb{F}_q$. Antag, at det ledende monomium af f er $x_1^{i_1} \cdots x_n^{i_n}$ hvor $0 \leq i_1, \ldots, i_n < q$. Så er antallet af ikke-nulpunkter for $f(x_1, \ldots, x_n)$ i $\mathbb{F}_q^m$ mindst $(q - i_1) \cdots (q - i_n)$.

Bevis:
Vi starter med at se, at

$$\begin{aligned}&\Delta_{>}(\langle x_1^q - x_1, \ldots, x_n^q - x_n, f(x_1, \ldots, x_n)\rangle)\\ \subseteq&\{x_1^{s_1} \cdots x_n^{s_n} \mid 0 \leq s_t < q \text{ for } t = 1, \ldots n \text{ og } i_t > s_t \text{ for mindst et } t = 1, \ldots, n\}\end{aligned} \tag{2.4}$$

Vi ved således fra sætning 2.44 at der må findes færre nulpunkter end størrelsen af mængden i ligningen (2.4). Der må derfor findes mindst lige så mange ikke-nulpunkter som i mængden

$$\{x_1^{s_1} \cdots x_n^{s_n} \mid 0 \leq s_t < q \text{ for } t = 1, \ldots n \text{ og } i_t \leq s_t \text{ for alle } t = 1, \ldots, n\}$$

Hvilket har en størrelse på $(q - i_1) \cdots (q - i_n)$, hvilket beviser sætningen.

■

Kapitel 3

Kötter-Kschischang koder

Vi bevæger os nu fra algebraen over til netværkskodning med Kötter-Kschischang koder, som er underrumskoder, og udvider lineære multicast netværkskodningsproblemer til nu også at indeholde det tilfælde, hvor der kan ske fejl og sletninger i netværket. Der kan altså forsvinde data eller data kan blive forvansket. Dette kapitel er baseret på en artikel af Kötter og Kschischang [12]. Kötter-Kschischang koder er en form for koder, som på sin vis minder en del om Reed-Solomon koder. Det er fejl- og sletningskorrigerende koder, som virker på operatorkanaler.

3.1 Operatorkanalen

Vi introducerer operatorkanalen for et enkelt unicast, altså en afsender til en modtager. Det er ligetil at generalisere til multicast, hvor der er en afsender og flere modtagere.
Vi genkalder hvordan tilfældig lineær netværkskodning fungerer. Kommunikation mellem afsender og modtager foregår i en række "runder", under hvilke afsenderen indskyder et antal pakker af fast længde ind i netværket. Hver af disse pakker kan anskues som rækkevektorer af længde N over et endeligt legeme $\mathbb{F}_q$. Disse pakker propagerer gennem netværket og passerer muligvis en række mellemliggende punkter på vejen mellem afsender og modtager. Hver gang et mellemliggende punkt har muligheden for at sende en pakke laver den en tilfældig $\mathbb{F}_q$-lineær kombination af de pakker den har til rådighed og sender dette. Dette betyder samtidig, at det samme mellemliggende punkt kan sende forskellige tilfældige kombinationer på forskellige kanter. Til slut samler modtageren alle de tilfældige pakker, som når frem til den og prøver at udlede

mængden af pakker, som oprindeligt blev sendt ind i netværket. Der er ingen antagelse om, at netværket skal være acyklisk.
Mængden af succesfulde pakketransmissioner (repræsenteret ved de kanter de bevægede sig over) i en runde inducerer en orienteret multigraf med den samme punktmængde som netværket, i hvilken kanterne repræsenterer succesfulde pakketransmissioner. Raten af informationstransmission (pakker per runde) mellem afsenderen og modtageren er øvrebegrænset af minimumscuttet mellem disse punkter, altså af det mindste antal kanter, som skal fjernes for at bryde forbindelsen mellem disse punkter. Det er kendt, at tilfældig lineær netværkskodning i $\mathbb{F}_q$ kan opnå en transmissionsrate, som tilsvarer minimumscuttet, med en sandsynlighed, der går mod en når q går mod uendelig [7], som vi også så i afsnit 1.3.
Lad $\{p_1, p_2, \ldots, p_M\}$, $p_i \in \mathbb{F}_q^N$, være mængden af indskudte vektorer. I det tilfælde, hvor der ikke sker fejl, får modtageren pakker $y_j, j = 1, 2, \ldots, L$, hvor hvert enkelt y_j er givet ved $y_j = \sum_{i=1}^{M} h_{j,i} p_i$ med ukendte, tilfældigt valgte koefficienter $h_{i,j} \in \mathbb{F}_q$.
Vi bemærker, at L ikke er fastsat fra starten, og normalt vil modtageren samle så mange pakker som muligt. Dog kan egenskaber i netværket, som f.eks. minimumscuttet mellem afsender og modtager, have indflydelse på fordelingen af $h_{i,j}$'erne og på et tidspunkt vil det ikke længere være gavnligt at indsamle flere pakker med redundant information.
Hvis vi vælger at betragte indskydelsen af T fejlagtige pakker, bliver denne model udvidet til at inkludere fejlpakker $e_t, t = 1, \ldots, T$ således, at

$$y_j = \sum_{i=1}^{M} h_{j,i} p_i + \sum_{t=1}^{T} g_{j,t} e_t,$$

hvor også $g_{j,t} \in \mathbb{F}_q$ er ukendte tilfældige koefficienter. Bemærk, at da disse fejlpakker kan indskydes hvor som helst i netværket, kan de give omfattende fejlpropagering. Specielt hvis $g_{j,1} \neq 0$ for alle j, har selv en enkelt fejlpakke e_1 potentialet til at korrumpere hver eneste modtaget pakke.
På matrixform kan transmissionsmodellen skrives som

$$y = Hp + Ge, \tag{3.1}$$

hvor H og G er hhv. en tilfældig $L \times M$- og en $L \times T$-matrix, p er den $M \times N$-matrix, hvis rækker er transmissionsvektorerne, y er den $L \times N$-matrix, hvis rækker er de modtagne vektorer og e er den $T \times N$-matrix, hvis rækker er fejlvektorerne.

Netværkets topologi vil ganske givet have indflydelse på strukturen af matricerne H og G. For eksempel kan H have ikke-fuld rang hvis minimumscuttet mellem afsender og modtager ikke er stort nok til at understøtte transmissionen af M uafhængige pakker i løbet af en runde. Selvom muligheden for at udnyttet netværkets struktur eksisterer, vil vi i dette kapitel ikke tage hensyn til den finere struktur af matricen H. Vi vil kunne fjerne en sådan finere struktur ved randomisering af kilde, f.eks. hvis afsenderen i stedet for at indskyde pakker p_i i netværket indskød tilfældige lineære kombinationer af p_i'erne.
Da H er tilfældig kan vi spørge os selv, hvilke egenskaber ved de indskudte vektorer vi bibeholder, når vi benytter kanalen beskrevet ved (3.1), selv hvis der ikke er noget støj ($e = 0$). Da H er tilfældig er det eneste, der er fast i produktet Hp rækkerummet af p. For så vidt gælder modtageren er alle mulige mængder, som genererer dette rum, ækvivalente. Dette fører til, at vi kan betragte informationstransmissionen, ikke i forhold til valget af p, men i forhold til valget af det vektorrum, der udspændes af rækkerne i p. Denne observation er grundstenen i den kanalmodel og de transmissionsstrategier, som vi vil betragte i dette kapitel. Hvis man betragter vektorrummet, der vælges af afsenderen, så er den eneste ødelæggende effekt en multiplikation med H kan have, at Hp har lavere rang end p, f.eks. pga. utilstrækkeligt minimumscut eller pakkesletninger, hvilket resulterer i, at Hp genererer et underrum af rækkerummet af p.
Lad W være et fast N-dimensionelt rum over $\mathbb{F}_q$. Alle transmitterede og modtagne pakker er vektorer i W, men vi vil beskrive transmissionsmodellen ud fra underrum af W, som udspændes af disse pakker. Lad $\mathcal{P}(W)$ være mængden af alle underrum af W. Dimensionen af et element $V \in \mathcal{P}(W)$ noteres med $\dim(V)$. Summen af to underrum U og V af W er $U + V = \{u + v \mid u \in U, v \in V\}$. Ækvivalent er $U + V$ det mindste underrum af W, som indeholder både U og V. Hvis $U \cap V = \{0\}$, så er summen $U + V$ en direkte sum, noteret $U \oplus V$. Det er klart, at $\dim(U \oplus V) = \dim(U) + \dim(V)$. For vilkårlige underrum U og V har vi $V = (U \cap V) \oplus V'$ for et underrum V', som er isomorft til kvotientrummet $V/(U \cap V)$. I dette tilfælde er $U + V = U + ((U \cap V) \oplus V') = U \oplus V'$.
Lad $\mathcal{P}(W, k)$ være mængden af alle underrum af W af dimension k.
Vi er nu klar til at definerer operatorkanalen.

Definition 3.1:
En operatorkanal C, som er tilknyttet det omsluttende rum W er en kanal med input- og outputalfabet $\mathcal{P}(W)$. Vi kan finde kanaloutput U ud fra

kanalinput V således, at

$$U = \omega \oplus E,$$

hvor $\omega \in \mathcal{P}(V, k)$, når $k = \dim(U \cap V)$ og $E \in \mathcal{P}(W)$ er et fejlrum. Når operatorkanalen omdanner V til U siger vi, at den begår $\rho = \dim(V) - k$ sletninger og $t = \dim(E)$ fejl.

Vi modellerer fejlrummet E således, at det intet har tilfælles med inputrummet V og valget af E er således ikke uafhængigt af V. Havde vi valgt at modellere outputrummet som $U = \omega + E$, for et arbitrært fejlrum E, så ville vi, idet $E = (E \cap V) \oplus E'$ for et rum E', have, at $U = \omega + (E \cap V) \oplus E' = \omega' \oplus E'$, hvor $\omega' \in \mathcal{P}(W, k')$, $k' > k$. Det vil altså sige, at den del af fejlrummet E, som var fælles med inputrummet V kun vil være gavnlig, og muligvis nedsætte antallet af sletninger hos modtageren.
Lad os opsummere. En operatorkanal tager altså et vektorrum ind og sender et andet vektorrum ud, muligvis med sletninger (fjernelse af vektorer fra det oprindelige rum) og/eller fejl (tilføjelse af nye vektorer til det transmitterede vektorrum).
Lad os nu fortsætte til Kötter-Kschischang koder.

3.2 Koder

Lad W være et N-dimensionelt vektorrum over $\mathbb{F}_q$. En kode for en operatorkanal med det omsluttende rum W er en ikke-tom delmængde af $\mathcal{P}(W)$, altså en ikke-tom mængde af underrum af W.
Størrelsen af en kode $\mathcal{K}$ noteres $|\mathcal{K}|$ og minimumsafstanden af $\mathcal{K}$ er givet ved

$$D(\mathcal{K}) := \min_{X,Y \in \mathcal{K}; X \neq Y} \{d(X, Y)\}.$$

Hvor $d(X, Y) = \dim(X) + \dim(Y) - 2\dim(X \cap Y)$. Den maksimale dimension af kodeordene i $\mathcal{K}$ er givet ved

$$l(\mathcal{K}) := \max_{X \in \mathcal{K}} \{\dim(X)\}.$$

Hvis dimensionen af alle kodeord i $\mathcal{K}$ er den samme så siges $\mathcal{K}$ at være en kode af konstant dimension.

Parallelt til den klassiske triple $[n, k, d]$ for en lineær fejlkorrigerende kode af længde n, dimension k og med minimums-Hammingdistance d, kan en kode $\mathcal{K}$ for en operatorkanal med et N-dimensionalt omsluttende rum over $\mathbb{F}_q$ siges at være af typen $[N, l(\mathcal{K}), \log_q |\mathcal{K}|, D(\mathcal{K})]$. Her er N længden af kodeordene, $l(\mathcal{K})$ er dimensionen af det rum, som vi skal sende, $\log_q |\mathcal{K}|$ er størrelsen af koden og $D(\mathcal{K})$ er minimumsafstanden. Det er altså $l(\mathcal{K})$, som er nyt i forhold til klassiske koder.
Inden vi kan introducere selve ind- og dekodningsmetoden skal vi først kende noget teori.

3.3 Lineariserede polynomier

Lad $\mathbb{F}_q$ være det endeligt legeme med q elementer og lad $\mathbb{F} = \mathbb{F}_{q^m}$ være et udvidelseslegeme. Et polynomium $L(x)$ kaldes et lineariseret polynomium over $\mathbb{F}$ hvis det er på formen

$$L(x) = \sum_{i=0}^{d} a_i x^{q^i},$$

hvor $a_i \in \mathbb{F}, i = 0, \ldots, d$. Når q er fast skriver vi $x^{[i]}$ i stedet for x^{q^i}. Det lineariserede polynomium er så

$$L(x) = \sum_{i=0}^{d} a_i x^{[i]}.$$

Hvis $L_1(x)$ og $L_2(x)$ er lineariserede polynomier over $\mathbb{F}$, så er enhver $\mathbb{F}$-lineær kombination $\alpha_1 L_1(x) + \alpha_2 L_2(x)$, $\alpha_1, \alpha_2 \in \mathbb{F}$ også et lineariseret polynomie. Det ordinære produkt $L_1(x)L_2(x)$ er ikke nødvendigvis et lineariseret polynomium. For kompositionen $L_1(L_2(x))$, som skrives $L_1(x) \otimes L_2(x)$ gælder følgende.

Sætning 3.2:
Givet lineariserede polynomier $L_1(x)$ og $L_2(x)$ så er $L_1(x) \otimes L_2(x)$ også et lineariseret polynomium.

Bevis:
Hvis $L_1(x) = \sum_{i\geq 0} a_i x^{[i]}$ og $L_2(x) = \sum_{j\geq 0} b_j x^{[j]}$ så gælder det, at

$$\begin{aligned} L_1(x) \otimes L_2(x) = L_1(L_2(x)) &= \sum_{i\geq 0} a_i (L_2(x))^{[i]} \\ &= \sum_{i\geq 0} a_i \left(\sum_{j\geq 0} b_j x^{[j]} \right)^{[i]} \\ &= \sum_{i\geq 0} \sum_{j\geq 0} a_i b_j^{[i]} x^{[i+j]} \\ &= \sum_{k\geq 0} c_k x^{[k]}, \end{aligned}$$

hvor

$$c_k = \sum_{i=0}^{k} a_i b_{k-i}^{[i]}$$

Vi har således vist at $L_1(x) \otimes L_2(x)$ er et lineariseret polynomium.

■

Som det ses af følgende eksempel er komposition ikke kommutativ.

Eksempel 3.3:
Hvis $q = 2$ og de lineariserede polynomier $L_1(x) = x^{[1]}$ og $L_2(x) = 2x^{[2]}$ så er

$$L_1(x) \otimes L_2(x) = \left(2x^{[2]}\right)^{[1]} = 4x^{[3]}$$

$$L_2(x) \otimes L_1(x) = 2\left(x^{[1]}\right)^{[2]} = 2x^{[3]}$$

🦋

Dette fører logisk til, at der må eksistere to forskellig måder at dividere på, nemlig fra højre eller venstre. Således gælder for to lineariserede polynomier $a(x)$ og $b(x)$, at der eksisterer unikke lineariserede polynomier $q_L(x)$, $q_R(x)$, $r_L(x)$ og $r_R(x)$, således, at

$$a(x) = q_L(x) \otimes b(x) + r_L(x) = b(x) \otimes q_R(x) + r_R(x),$$

hvor $r_L \equiv 0$ eller $\deg(r_L(x)) < \deg(b(x))$ og på samme måde $r_R \equiv 0$ eller $\deg(r_R(x)) < \deg(b(x))$. Vi kan let bestemme polynomierne $q_R(x)$ og $r_R(x)$ ved hjælp af algoritme 4, som er en adaption af den almindelige algoritme for polynomiel division.

Algoritme 4: RDIV(a(x),b(x))

Input: et par af lineariserede polynomier $a(x)$ og $b(x)$ over $\mathbb{F}_{q^m}$, hvor $b(x) \not\equiv 0$
Output: et par $q(x), r(x)$ af lineariserede polynomier over $\mathbb{F}_{q^m}$
$q(x) = 0$, $r(x) = 0$
$e = \deg(b(x))$, $d = \deg(a(x))$
while $d \geq e$ **do**
  $a_d = LC(a(x))$, $b_e = LC(b(x))$
  $t(x) = (a_d/b_e)^{[m-e]} x^{[d-e]}$
  $q(x) = q(x) + t(x)$
  $a(x) = a(x) - b(x) \otimes t(x)$
  $d = \deg(a(x))$
end
$r(x) = a(x)$
return $q(x), r(x)$

Algoritme 4 returnerer to polynomier $q(x)$ og $r(x)$, som opfylder, at $a(x) = b(x) \otimes q(x) + r(x)$, hvor enten $r(x) \equiv 0$ eller $\deg(r(x)) < \deg(b(x))$. Proceduren til venstredivision er den samme, men inde i while-loopet opdaterer vi i stedet således, at $t(x) = \left(a_d / \left(b_e^{[d-e]}\right)\right) x^{[d-e]}$ og $a(x) = a(x) - t(x) \otimes b(x)$. Dette resulterer i to polynomier $q(x)$ og $r(x)$, som opfylder, at $a(x) = q(x) \otimes b(x) + r(x)$.
Der gælder desuden følgende egenskab for lineariserede polynomier, hvilket giver dem deres navn.

Sætning 3.4:
Lad $L(x)$ være et lineariseret polynomium over $\mathbb{F} = \mathbb{F}_{q^m}$ og lad K være et vilkårligt udvidelseslegeme over $\mathbb{F}$. Vi kan anse K for værende et vektorrum over $\mathbb{F}_q$. Afbildningen, der tager $\beta \in K$ over i $L(\beta) \in K$ er lineær i forhold til $\mathbb{F}_q$, hvilket vil sige, at for alle $\beta_1, \beta_2 \in K$ og alle $\lambda_1, \lambda_2 \in \mathbb{F}_q$ har vi, at

$$L(\lambda_1\beta_1 + \lambda_2\beta_2) = \lambda_1 L(\beta_1) + \lambda_2 L(\beta_2). \tag{3.2}$$

Bevis:
Vi omregner i ligningen.

$$\begin{aligned} L(\lambda_1\beta_1 + \lambda_2\beta_2) &= \sum_{i\geq 0} c_i(\lambda_1\beta_1 + \lambda_2\beta_2)^{q^i} \\ &= \sum_{i\geq 0} c_i \left(\sum_{j=0}^{q^i} \binom{q^i}{j} (\lambda_1\beta_1)^j (\lambda_2\beta_2)^{q^i-j} \right) \\ &= \sum_{i\geq 0} c_i \left(\sum_{j=0}^{q^i} \frac{q^i!}{j!(q^i-j)!} (\lambda_1\beta_1)^j (\lambda_2\beta_2)^{q^i-j} \right). \end{aligned}$$

Når $d_{i,j} = \frac{q^i!}{j!(q^i-j)!}$ ses som værende et element i $\mathbb{F}$ er $d_{i,j} = 0$, da $d_{i,j}$ er et multiplum af q^i bortset fra situationen hvor $j = q^i$ eller $j = 0$. I disse to tilfælde har vi $d_{i,q^i} = d_{i,0} = 1$. Således får vi

$$\begin{aligned} L(\lambda_1\beta_1 + \lambda_2\beta_2) &= \sum_{i\geq 0} c_i \left(\sum_{j=0}^{q^i} \frac{q^i!}{j!(q^i-j)!} (\lambda_1\beta_1)^j (\lambda_2\beta_2)^{q^i-j} \right) \\ &= \sum_{i\geq 0} c_i \left((\lambda_1\beta_1)^{q^i} + (\lambda_2\beta_2)^{q^i} \right) \\ &= \sum_{i\geq 0} c_i (\lambda_1\beta_1)^{q^i} + \sum_{i\geq 0} c_i (\lambda_2\beta_2)^{q^i} \\ &= \sum_{i\geq 0} c_i \lambda_1 \beta_1^{q^i} + \sum_{i\geq 0} c_i \lambda_2 \beta_2^{q^i}, \end{aligned}$$

hvor vi bruger, at for $\lambda \in F_q$ gælder, at $\lambda^{q-1} = 0$ og altså $\lambda^q = \lambda$. Det giver, at $\lambda^{q^i} = (\lambda^q)^{q^{i-1}} = \lambda^{q^{i-1}} = (\lambda^q)^{q^{i-2}} = \cdots = \lambda$. Videre er

$$\begin{aligned} L(\lambda_1\beta_1 + \lambda_2\beta_2) &= \sum_{i\geq 0} c_i \lambda_1 \beta_1^{q^i} + \sum_{i\geq 0} c_i \lambda_2 \beta_2^{q^i} \\ &= \lambda_1 \sum_{i\geq 0} c_i \beta_1^{q^i} + \lambda_2 \sum_{i\geq 0} c_i \beta_2^{q^i} \\ &= \lambda_1 L(\beta_1) + \lambda_2 L(\beta_2), \end{aligned}$$

hvilket viser sætningen.

■

Følgende lemma viser, at alle linearkombinationer af nulpunkter også er nulpunkter.

Lemma 3.5:
Hvis $h(x)$ er et lineariseret polynomium og $h(\alpha_1) = h(\alpha_2) = \cdots = h(\alpha_{|A|}) = 0$ så er $h\left(\sum_{i=1}^{|A|} c_i\alpha_i\right) = 0$

Bevis:
Givet et lineariseret polynomium $h(x)$ hvor $h(\alpha_1) = h(\alpha_2) = \cdots = h(\alpha_{|A|}) = 0$ så er

$$\begin{aligned} h\left(\sum_{i=1}^{|A|} c_i\alpha_i\right) &= \sum_{i=1}^{|A|} h\left(c_i\alpha_i\right) \\ &= \sum_{i=1}^{|A|} c_i h\left(\alpha_i\right) \\ &= \sum_{i=1}^{|A|} c_i 0 \\ &= 0 \end{aligned}$$

Omskrivningerne følger af sætning 3.4.

■

Følgende lemma viser, at hvis to lineariserede polynomier af grad mindre end q^{d-1} er har samme værdi i d lineært uafhængige punkter, så er de to polynomier kongruente.

Lemma 3.6:
Lad d være et positivt heltal og lad $f(x)$ og $g(x)$ være to lineariserede polynomier over $\mathbb{F} = \mathbb{F}_{q^m}$ af grad mindre end q^d. Hvis $\alpha_1, \ldots, \alpha_d$ er lineært uafhængige elementer i K, hvor K er et vilkårligt udvidelseslegeme af $\mathbb{F}$, således, at $f(\alpha_i) = g(\alpha_i)$ for $i = 1, \ldots, d$, så er $f(x) \equiv g(x)$.

Bevis:
Observer, at $h(x) = f(x) - g(x)$ har $\alpha_1, \ldots, \alpha_d$ som nulpunkter og derfor også pr. lemma 3.5 alle q^d linear kombinationer af disse som nulpunkter. Altså har

$h(x)$ mindst q^d forskellige nulpunkter. Men da graden af $h(x)$ er mindre end q^d, er dette kun muligt hvis $h(x) \equiv 0$, hvorfor $f(x) \equiv g(x)$.

■

Lemma 3.7:
Givet to lineariserede polynomier $f(x)$ og $g(x)$ så er $h(x) = f(x) - g(x)$ også et lineariseret polynomium.

Bevis:
Givet to lineariserede polynomier $f(x) = \sum_{i=0}^{k} a_i x^{[i]}$ og $g(x) = \sum_{j=0}^{l} b_j x^{[j]}$ kan vi uden tab af generalitet antage at $k \geq l$. Vi har da, at

$$\begin{aligned} h(x) &= f(x) - g(x) \\ &= \sum_{i=0}^{k} a_i x^{[i]} - \sum_{j=0}^{l} b_j x^{[j]} \\ &= \sum_{i=0}^{k} c_i x^{[i]} \end{aligned}$$

Hvor $c_i = \begin{cases} a_i - b_i & \text{hvis } i \leq l \\ a_i & \text{ellers} \end{cases}$

■

3.4 Konstruktion af koder

Lad os nu betragte konstruktionen af en kode. Lad $\mathbb{F}_q$ være et endeligt legeme og $\mathbb{F} = \mathbb{F}_{q^m}$ være et (endeligt) udvidelseslegeme over $\mathbb{F}_q$. Vi betragter $\mathbb{F}$ som værende et vektorrum af dimension m over $\mathbb{F}_q$. Lad $A = \{\alpha_1, \ldots \alpha_l\} \subseteq \mathbb{F}$ være er mængde af lineært uafhængige elementer i dette vektorrum. Disse elementer udspænder et l-dimensionalt vektorrum $\langle A \rangle \subseteq \mathbb{F}$ over $\mathbb{F}_q$. Det er klart, at $l \leq m$. Vi tager som vores omsluttende rum den direkte sum $W = \langle A \rangle \oplus \mathbb{F} = \{(\alpha, \beta) \mid \alpha \in \langle A \rangle, \beta \in \mathbb{F}\}$, som er et vektorrum af dimension $l + m$ over $\mathbb{F}_q$. Lad $u = (u_0, u_1, \ldots, u_{k-1}) \in \mathbb{F}^k$ betegne en blok af beskedsymboler, der består af k symboler over $\mathbb{F}$ eller, ligedan, mk symboler over $\mathbb{F}_q$. Lad $\mathbb{F}^k[x]$ betegne mængden af lineariserede polynomier over $\mathbb{F}$ af grad højst q^{k-1}. Lad

$f(x) \in \mathbb{F}^k[x]$, som er defineret ved

$$f(x) = \sum_{i=0}^{k-1} u_i x^{[i]},$$

være det lineariserede polynomium med koefficienter, der svarer til u. Lad, som det sidste, $\beta_i = f(\alpha_i)$. Ethvert af parrene $(\alpha_i, \beta_i), i = 1, \ldots, l$ kan anses som værende vektorer i W. Bemærk her, at både α'erne og β'erne har længde m, således, at længden af den samlede vektor bliver $2m$, det er dog kun nødvendigt, at benytte en vektor af længde $l+m$, som vi så, for rummet W. Dette betyder, at vi kan ændre vores α'er til vektorer af længde l inden vi sender dem. Da $\langle A \rangle$ er et l-dimensionelt rum er det stadig muligt at vælge l lineært uafhængige α'er. Idet $\{\alpha_1, \ldots, \alpha_l\}$ er en lineært uafhængig mængde, så er $\{(\alpha_1, \beta_1), \ldots, (\alpha_l, \beta_l)\}$ det også. Denne mængde udspænder altså et l-dimensionelt underrum V af W. Vi kalder den afbildning, der tager besked polynomiet $f(x) \in \mathbb{F}^k[x]$ over i det lineære rum $V \in \mathcal{P}(W, |A|)$ for $\mathbf{ev}_A$. Vi har altså $\mathbf{ev}_A : \mathbb{F}^k[x] \to \mathcal{P}(W, |A|)$, hvilket betyder, at $\mathbf{ev}_A(f(x)) = Span_{\mathbb{F}}\{(\alpha_1, f(\alpha_1)), \ldots, (\alpha_l, f(\alpha_l))\}$.

Lemma 3.8:
Hvis $|A| \geq k$, så er afbildningen $\mathbf{ev}_A : \mathbb{F}^k[x] \to \mathcal{P}(W, |A|)$ injektiv.

Bevis:
Antag $|A| \geq k$ og $\mathbf{ev}_A(f(x)) = \mathbf{ev}_A(g(x))$ for to funktioner $f(x), g(x) \in \mathbb{F}^k[x]$. Lad $h(x) = f(x) - g(x)$. Vi har

$$\mathbf{ev}_A(f(x)) = Span_{\mathbb{F}}\{(\alpha_1, f(\alpha_1)), \ldots, (\alpha_{|A|}, f(\alpha_{|A|}))\}$$

og

$$\mathbf{ev}_A(g(x)) = Span_{\mathbb{F}}\{(\alpha_1, g(\alpha_1)), \ldots, (\alpha_{|A|}, g(\alpha_{|A|}))\},$$

hvorom det gælder, at alle $(\alpha_i, f(\alpha_i))$ er lineært uafhængige og alle $(\alpha_i, g(\alpha_i))$ er lineært uafhængige. Da $\mathbf{ev}_A(f) = \mathbf{ev}_A(g)$, så må det gælde, at $(\alpha_i, f(\alpha_i)) = (\alpha_i, g(\alpha_i))$ for alle $i = 0, \ldots, |A|$. Det betyder altså, at $h(\alpha_i) = 0$ for alle $i = 0, \ldots, |A|$.
Vi ved fra lemma 3.7, at $h(x)$ er et lineariseret polynomium og fra lemma 3.5 ved vi, at $h(x) = 0$ for alle $x \in \langle A \rangle$. Det vil altså sige, at $h(x)$ har mindst $q^{|A|} \geq q^k$ nulpunkter. Da $h(x)$ har grad højst q^{k-1} er dette kun muligt hvis $h(x) \equiv 0$. Dette betyder, at $f(x) \equiv g(x)$ og dermed er $\mathbf{ev}_A$ injektiv.

■

Vi vil fra nu af antage, at $l = |A| \geq k$. Vi har da fra lemma 3.8, at afbildningen af $\mathbb{F}^k[x]$ er en kode $\mathcal{K} \subseteq \mathcal{P}(W, l)$ med q^{mk} kodeord.
Vi ønsker nu at bestemme minimumsafstanden af $\mathcal{K}$ og indfører derfor følgende lemma.

Lemma 3.9:
Hvis $\{(\alpha_1, \beta_1), \ldots, (\alpha_r, \beta_r)\} \subseteq W$ er en samling af r lineært uafhængige elementer, der opfylder, at $\beta_i = f(\alpha_i)$ for et lineariseret polynomium f over $\mathbb{F}$, så er $\{\alpha_1, \ldots, \alpha_r\}$ en lineært uafhængig mængde.

Bevis:
Vi har givet, at $\{(\alpha_1, \beta_1), \ldots, (\alpha_r, \beta_r)\} \subseteq W$ er en samling af r lineært uafhængige elementer, der opfylder, at $\beta_i = f(\alpha_i)$ for et lineariseret polynomium f over $\mathbb{F}$.
Antag nu, at $\{\alpha_1, \ldots, \alpha_r\}$ er en lineært afhængig mængde. Vi ved da, at det for nogle $\gamma_1, \ldots, \gamma_r \in \mathbb{F}_q$, som ikke alle er 0, gælder, at $\sum_{i=1}^{r} \gamma_i \alpha_i = 0$.
Vi vil da i W have, at

$$\begin{aligned}
\sum_{i=1}^{r} \gamma_i(\alpha_i, \beta_i) &= \left(\sum_{i=1}^{r} \gamma_i \alpha_i, \sum_{i=1}^{r} \gamma_i \beta_i \right) \\
&= \left(0, \sum_{i=1}^{r} \gamma_i f(\alpha_i) \right) \\
&= \left(0, f\left(\sum_{i=1}^{r} \gamma_i \alpha_i \right) \right) \\
&= (0, f(0)) \\
&= (0, 0)
\end{aligned}$$

fordi f er et lineariseret polynomium.
Da (α_i, β_i)-parrene er lineært uafhængige kan dette kun lade sig gøre, hvis $\gamma_1, \ldots, \gamma_r = 0$, hvilket er i modstrid med antagelsen om, at $\{\alpha_1, \ldots, \alpha_r\}$ en lineært afhængig mængde, hvorfor den må være lineært uafhængig.

■

Lad os nu definere koden $\mathcal{K}$ fuldstændigt.

Sætning 3.10:
Lad $\mathcal{K}$ være afbildningen af $\mathbb{F}^k[x]$ under $\mathbf{ev}_A$, med $l = |A| \geq k$. Så er $\mathcal{K}$ en kode af typen $[l+m, l, mk, 2(l-k+1)]$.

Bevis:
Det ses let, at længden er $l+m$, dimensionen af det sendte rum er l og størrelsen af koden er ifølge lemma 3.8 mk. Det eneste vi mangler er altså at vise minimumsafstanden.
Lad $f(x)$ og $g(x)$ være to forskellige elementer i $\mathbb{F}^k[x]$ og lad $U = \mathbf{ev}_A(f(x))$ og $V = \mathbf{ev}_A(g(x))$. Vi siger, at $U \cap V$ har dimension r. Det betyder, at det er muligt at finde r lineært uafhængige elementer $(\alpha'_1, \beta'_1), \ldots, (\alpha'_r, \beta'_r)$ således, at $f(\alpha'_i) = g(\alpha'_i) = \beta'_i$ for $i = 1, \ldots, r$. Af lemma 3.9 ses, at $(\alpha'_1, \ldots, \alpha'_r)$ er lineært uafhængige og de udspænder dermed et r-dimensionelt rum B, hvori det gælder, at $f(b) - g(b) = 0$ for alle $b \in B$. Hvis $r \geq k$ så ville $f(x)$ og $g(x)$ være to lineariserede polynomier af grad mindre end q^k, som var ens i mindst k lineært uafhængige punkter og vi har da fra lemma 3.6 at $f(x) \equiv g(x)$. Da dette ikke er tilfældet må vi have, at $r \leq k-1$. Altså har vi

$$\begin{aligned} d(U,V) &= dim(U) + dim(V) - 2dim(U \cap V) \\ &= l + l - 2r \\ &= 2l - 2r \\ &\geq 2l - 2(k-1) \\ &= 2(l-k+1). \end{aligned}$$

Der kan gælde lighed hvis $g(x)$ er nulpolynomiet og $f(x)$ er et polynomie af grad q^{k-1}. De kan da have $k-1$ nulpunkter til fælles uden at $f(x) \equiv g(x)$.
Vi har nu vist, at $\mathcal{K}$ er en kode af typen $[l+m, l, mk, 2(l-k+1)]$.

■

Lad os nu betragte et par eksempler på hvordan man indkoder med Kötter-Kschischang koder.

Eksempel 3.11:
Vi vælger det endelige legeme $\mathbb{F}_2$ og lader $\mathbb{F} = \mathbb{F}_{2^4}$ være det endelige udvidelseslegeme af $\mathbb{F}_2$. Vi repræsenterer $\mathbb{F}_{2^4} = \mathbb{F}_{16}$ ved $\mathbb{F}_2[s]/\langle s^4 + s + 1\rangle$, hvilket betyder, at $2 = 0$ og $1 + s + s^4 = 0 \Leftrightarrow s^4 = 1 + s$.
Legemet $\mathbb{F}$ kan anskues som et vektorrum af dimension 4 over F_2, hvilket betyder, at vi kan repræsentere $\mathbb{F}$ på to måder - enten som en mængde af

polynomier eller som en mængde af vektorer, som det passer til vores formål.

$$\begin{aligned}\mathbb{F} &= \{a_0 + a_1 s + a_2 s^2 + a_3 s^3 \mid a_i \in \mathbb{F}_2\} \\ &= \{(a_0, a_1, a_2, a_3) \mid a_i \in \mathbb{F}_2\}\end{aligned}$$

Vi vælger

$$\begin{aligned}A &= \{\alpha_1, \alpha_2, \alpha_3, \alpha_4\} \\ &= \{1 + s^2, s^2 + s^3, s, 1 + s + s^2 + s^3\} \\ &= \{(1,0,1,0), (0,0,1,1), (0,1,0,0), (1,1,1,1)\}\end{aligned}$$

som vores mængde af lineært uafhængige elementer fra $\mathbb{F}$. Mængden A udspænder et 4-dimensionelt vektorrum $\langle A \rangle \subseteq \mathbb{F}$ over $\mathbb{F}_2$.
Som vores omsluttende rum W tager vi den direkte sum

$$\begin{aligned}W &= \langle A \rangle \oplus \mathbb{F} \\ &= \{(\alpha, \beta) \mid \alpha \in \langle A \rangle, \beta \in \mathbb{F}\}.\end{aligned}$$

Vi vælger nu vores besked u, som er 3 elementer over $\mathbb{F}$.

$$\begin{aligned}u &= (u_0, u_1, u_2) \\ &= (s^3, s + s^2 + s^3, 1 + s^2 + s^3) \\ &= ((0001), (0111), (1011)).\end{aligned}$$

Vi bruger funktionen

$$\begin{aligned}f(x) &= \sum_{i=0}^{2} u_i x^{2^i} \\ &= (0001)x + (0111)x^2 + (1011)x^4 \\ &= (s^3)x + (s + s^2 + s^3)x^2 + (1 + s^2 + s^3)x^4.\end{aligned}$$

Vi kan nu beregne $\beta_i = f(\alpha_i)$ for $i = 1, \ldots, 4$, men lad os først lave tabel 3.1 over reduktioner af højere grader af s. Husk, $2 = 0$ og $1+s+s^4 = 0 \Leftrightarrow s^4 = 1+s$. Husk også, at i $\mathbb{F}_q$ har vi $x^q - x = 0$, så hvis $x \neq 0$ har vi $x^{q-1} - 1 = 0 \Leftrightarrow x^{q-1} = 1$. Det betyder i vores tilfælde i $\mathbb{F}_{16}$ at $s^{15} = 1$ som det ses i tabeloversattelse. Dette betyder altså, at vi kan omskrive

$$f(x) = (s^3)x + (s + s^2 + s^3)x^2 + (1 + s^2 + s^3)x^4 = (s^3)x + (s^{11})x^2 + (s^{13})x^4.$$

$$
\begin{aligned}
s^4 &= 1 + s \\
s^5 &= s + s^2 \\
s^6 &= s^2 + s^3 \\
s^7 &= 1 + s + s^3 \\
s^8 &= 1 + s^2 \\
s^9 &= s + s^3 \\
s^{10} &= 1 + s + s^2 \\
s^{11} &= s + s^2 + s^3 \\
s^{12} &= 1 + s + s^2 + s^3 \\
s^{13} &= 1 + s^2 + s^3 \\
s^{14} &= 1 + s^3 \\
s^{15} &= 1
\end{aligned}
$$

Tabel 3.1: *Omskrivningstabel*

Nu er vi klar til at beregne β'erne.

$$
\begin{aligned}
\beta_1 = f(\alpha_1) &= f(1 + s^2) = f(s^8) \\
&= s^3 s^8 + s^{11}(s^8)^2 + s^{13}(s^8)^4 \\
&= s^{11} + s^{11} s^{16} + s^{13} s^{32} \\
&= s^{11} + s^{11} s + s^{13} s^2 \\
&= s^{11} + s^{12} + s^{15} \\
&= (s + s^2 + s^3) + (1 + s + s^2 + s^3) + 1 \\
&= 0 = (0, 0, 0, 0) \\
\beta_2 = f(\alpha_2) &= f(s^2 + s^3) = f(s^6) \\
&= s^3 s^6 + s^{11}(s^6)^2 + s^{13}(s^6)^4 \\
&= s^9 + s^8 + s^7 \\
&= (s + s^3) + (1 + s^2) + (1 + s + s^3) \\
&= s^2 = (0, 0, 1, 0)
\end{aligned}
$$

$$
\begin{aligned}
\beta_3 &= f(\alpha_3) = f(s) \\
&= s^3 s + s^{11} s^2 + s^{13} s^4 \\
&= s^4 + s^{13} + s^2 \\
&= (1+s) + (1+s^2+s^3) + s^2 \\
&= s + s^3 = (0,1,0,1) \\
\beta_4 &= f(\alpha_4) = f(1+s+s^2+s^3) = f(s^{12}) \\
&= s^3 s^{12} + s^{11}(s^{12})^2 + s^{13}(s^{12})^4 \\
&= s^{15} + s^5 + s \\
&= 1 + (s+s^2) + s \\
&= 1 + s^2 = (1,0,1,0)
\end{aligned}
$$

Mængden

$$
\{(\alpha_1,\beta_1),\ldots,(\alpha_4,\beta_4)\} =
$$
$$
\{(1,0,1,0;0,0,0,0),(0,0,1,1;0,0,1,0),(0,1,0,0;0,1,0,1),(1,1,1,1;1,0,1,0)\}
$$

er lineært uafhængig og udspænder et 4-dimensionelt underrum V af W. Underrummet V er vores indkodning.

Lad os gentage det foregående eksempel, men denne gang vælge et mindre underrum.

Eksempel 3.12:
Vi vælger igen det endelige legeme $\mathbb{F}_2$ og lader $\mathbb{F} = \mathbb{F}_{2^4}$ være et endeligt udvidelseslegeme af $\mathbb{F}_2$. Vi repræsenterer stadig $\mathbb{F}_{2^4} = \mathbb{F}_{16}$ ved $\mathbb{F}_2[s]/\langle s^4+s+1\rangle$, hvilket igen betyder, at $2 = 0$ og $1 + s + s^4 = 0 \Leftrightarrow s^4 = 1 + s$.
Denne gang vælger vi

$$
\begin{aligned}
A &= \{\alpha_1, \alpha_2, \alpha_3\} \\
&= \{1+s^2, s^2+s^3, s\} \\
&= \{(1,0,1,0),(0,0,1,1),(0,1,0,0)\}
\end{aligned}
$$

som vores mængde af lineært uafhængige elementer af $\mathbb{F}$. Mængden A udspænder nu et 3-dimensionelt vektorrum $\langle A\rangle \subseteq \mathbb{F}$ over $\mathbb{F}_2$ i stedet for et 4-dimensionelt vektorrum. Vi benytter de samme α'er, bortset fra, vi kun bruger

de første tre.
Vi konstruerer vores omsluttende rum W på samme måde som før, og vælger den samme besked u som før. Vi bruger stadig funktionen

$$f(x) = (s^3)x + (s^{11})x^2 + (s^{13})x^4.$$

Da vi har valgt de samme α'er, beskeder og funktion er vores β'er de samme som før.

$$\begin{aligned}\beta_1 &= 0 = (0,0,0,0)\\ \beta_2 &= s^2 = (0,0,1,0)\\ \beta_3 &= s + s^3 = (0,1,0,1)\end{aligned}$$

Mængden

$$\begin{aligned}&\{(\alpha_1,\beta_1),(\alpha_2,\beta_2),(\alpha_3,\beta_3)\} =\\ &\{(1,0,1,0;0,0,0,0),(0,0,1,1;0,0,1,0),(0,1,0,0;0,1,0,1)\}\end{aligned}$$

er lineært uafhængig og udspænder et 3-dimensionelt underrum V af W, hvilket er vores indkodning. Vi kan dog i dette tilfælde reducere længden af vektorerne i denne mængde. Da $W = \langle A\rangle \oplus \mathbb{F}$ og $\langle A\rangle$ er et 3-dimensionelt vektorrum kan vi udspænde dette rum med tre vektorer af længde 3+4. Vi vælger vektorerne

$$\{(1,0,0;0,0,0,0),(0,1,0;0,0,1,0),(0,0,1;0,1,0,1)\}$$

i stedet.

3.5 Dekodning

Lad os nu betragte hvordan vi kan dekode det sendte.
Hvis der er sendt et l-dimensionelt rum $V \in \mathcal{K}$, som er et underrum af W, ind i operatorkanalen og der kommer et $(l-\rho+t)$-dimensionelt underrum U af W ud, så har vi, at $\rho = \dim(V) - \dim(U \cap V)$ er antallet af sletninger og $t = \dim(U) - \dim(U \cap V)$ er antallet af fejl. Vi forstår her en sletning, som en manglende dimension af det oprindelige l-dimensionelle rum V. Det betyder, at selvom U også er et l-dimensionelt rum, så kan der stadig godt være sletninger, hvis det ikke er det samme rum. Fejl forstår her som antallet af lineært uafhængige vektorer, der ikke ligger i V.

For dette underrum må der gælde, at $\dim(U \cap V) = (l - \rho)$ og ligeledes må der gælde $d(U, V) = \dim(U) + \dim(V) - 2\dim(U \cap V) = \rho + t$. Vi forventer at være i stand til at dekode hvis $\rho + t < D(\mathcal{K})/2 = l - k + 1$. Da der er indkodet k beskeder skal U mindst være et k-dimensionelt rum. Vi vil i det følgende beskrive hvordan man kan dekode en besked. Vi beskriver først det teoretiske grundlag for dekodningen og derefter hvordan det gøres i praksis.

Lad $r = l - \rho + t$ være dimensionen af det modtagne rum U og lad $(x_1, y_1), (x_2, y_2), \ldots, (x_r, y_r)$ være en basis for U, vi ønsker nu at kunne finde et bivariat lineariseret polynomium $Q(x, y)$ på formen $Q(x, y) = Q_x(x) + Q_y(y)$ hvor $Q(x_i, y_i) = 0$ for alle $i = 1, \ldots, r$, og vi har, at $Q_x(x)$ er et lineariseret polynomium over $\mathbb{F}_{q^m}$ af grad højest $q^{\tau-1}$ og $Q_y(y)$ er lineariseret polynomium af grad højest $q^{\tau-k}$. Da $Q(x, y)$ er et bivariat lineariseret polynomium vil det evaluerer til 0 i hele U og ikke kun på den valgte basis.

Lighederne $Q(x_i, y_i) = 0$ medfører, at vi har et system af r ligheder med $2\tau - k + 1$ ubekendte som har en ikke-triviel løsning når

$$r = l - \rho + t < 2\tau - k + 1 \tag{3.3}$$

Da $f(x)$ (fra indkodningen) er et lineariseret polynomium over $\mathbb{F}_{q^m}$ er $Q(x, f(x))$ det også. Polynomiet $Q(x, f(x))$ er givet ved

$$Q(x, f(x)) = Q_x(x) + Q_y(f(x)) = Q_x(x) + Q_y(x) \otimes f(x) \tag{3.4}$$

Da graden af $f(x)$ er højest q^{k-1} medfører det, at $Q(x, y)$ har grad højest $q^{\tau-1}$. Lad $\{(a_1, b_1), (a_2, b_2), \ldots, (a_{l-\rho}, b_{l-\rho})\}$ være en basis for $U \cap V$. Da disse vektorer alle ligger i U har vi at $Q(a_i, b_i) = 0$ for $i = 1, \ldots, l - \rho$. Da vektorerne ligeledes ligger i V gælder det, at $b_i = f(a_i)$ for $i = 1, \ldots, l - \rho$ og dermed

$$Q(a_i, b_i) = Q(a_i, f(a_i)) = 0 \quad \text{for } i = 1, \ldots, l - \rho$$

dermed er $Q(x, f(x))$ et lineariseret polynomium med $a_1, \ldots, a_{l-\rho}$ som nulpunkter. Fra lemma 3.9 ved vi at disse nulpunkter er lineært uafhængige, dermed er $Q(x, f(x))$ et lineariseret polynomie af grad højest $q^{\tau-1}$ som evaluerer til nul på et rum af dimension $l - \rho$. Hvis der gælder, at

$$l - \rho \geq \tau$$

så har $Q(x, f(x))$ flere nulpunkter end dets grad hvilket kun kan lade sig gøre hvis $Q(x, f(x)) = 0$. Er dette tilfældet må der gælde, at

$$\begin{aligned} Q(x, f(x)) &= 0 \\ Q_x(x) + Q_y(x) \otimes f(x) &= 0 \\ Q_y(x) \otimes f(x) &= -Q_x(x) \end{aligned}$$

Det vil nu være muligt at finde frem til $f(x)$ ved at bruge algoritme 4 med $(-Q_x(x), Q_y(x))$ som input.

Polynomiet $Q(x, y)$ kan findes ud fra de r basisvektorer for U. Polynomiet findes ved interpolation og vi vil beskrive en algoritme til effektivt at finde dette.
Først skal vi dog vide hvad den $(1, k-1)$-vægtede grad af et lineariseret polynomium er, denne er defineret på følgende måde:

$$\deg_{1,k-1}(f(x, y)) = \max\{d_x(f), k - 1 + d_y(f)\}$$

hvor $f(x) = f_x(x) + f_y(y)$ og graden af $f_x(x)$ og $f_y(y)$ er henholdsvis $q^{d_x(f)}$ og $q^{d_y(f)}$. Lad os betragte et eksempel på vægtet grad.

Eksempel 3.13:
Lad $q = 2$, $k = 3$ og $f(x, y) = x^{2^3} + x^{2^0} + y^{2^2}$ så er

$$\deg_{1,2}(f(x, y)) = \max\{3, 3 - 1 + 2\} = \max\{3, 4\} = 4.$$

I de tilfælde hvor der for eksempel ikke er en y del af polynomiet bliver y graden $-\infty$ som det ses her.
Lad $g(x, y) = x^{2^3} + x^{2^0}$ så er

$$\deg_{1,2}(g(x, y)) = \max\{3, -\infty\} = 3.$$

Algoritme 5 giver os et $Q(x, y)$ som opfylder kriterierne ovenfor.

Vi vil nu bevise, at det $Q(x, y)$ som algoritmen returnerer, opfylder de krav vi har sat til det, nemlig, at det evaluerer til nul på en basis for $U \cap V$ og, at det har grad lavere end $l - \rho$.

Vi definer en ordning $\prec$ på bivariate lineariserede polynomier som følgende.

Algoritme 5: Interpoler(U)

Input: En basis $(x_i, y_i) \in W, i = 1, \ldots, r$ for U
Output: Et lineariseret polynomium $Q(x, y) = Q_x(x) + Q_y(y)$
$f_0(x, y) = x \quad f_1(x, y) = y$
for $i = 1$ *to* r **do**
  $\Delta_0 = f_0(x_i, y_i) \quad \Delta_1 = f_1(x_i, y_i)$
  if $\Delta_0 = 0$ **then**
    $f_1(x, y) = f_1^q(x, y) - \Delta_1^{q-1} f_1(x, y)$
  else if $\Delta_1 = 0$ **then**
    $f_0(x, y) = f_0^q(x, y) - \Delta_0^{q-1} f_0(x, y)$
  else
    if $\deg_{1,k-1}(f_0) \leq \deg_{1,k-1}(f_1)$ **then**
      $f_1(x, y) = \Delta_1 f_0(x, y) - \Delta_0 f_1(x, y)$
      $f_0(x, y) = f_0^q(x, y) - \Delta_0^{q-1} f_0(x, y)$
    else
      $f_0(x, y) = \Delta_1 f_0(x, y) - \Delta_0 f_1(x, y)$
      $f_1(x, y) = f_1^q(x, y) - \Delta_1^{q-1} f_1(x, y)$
    end
  end
end
if $\deg_{1,k-1}(f_1) < \deg_{1,k-1}(f_0)$ **then**
  return $f_1(x, y)$
else
  return $f_0(x, y)$
end

Definition 3.14:
Vi siger, at $f(x,y) \prec g(x,y)$ *hvis*

$$\deg_{1,k-1}(f(x,y)) < \deg_{1,k-1}(g(x,y))$$

eller hvis

$$\deg_{1,k-1}(f(x,y)) = \deg_{1,k-1}(g(x,y))$$

og

$$d_y(f) + k - 1 < \deg_{1,k-1}(f(x,y)) \tag{3.5}$$

og

$$d_y(g) + k - 1 = \deg_{1,k-1}(g(x,y)). \tag{3.6}$$

Det betyder, at ved ens grad i de to polynomier, hvor $f(x,y) \prec g(x,y)$, får f sin grad fra x og g sin grad fra y.

Hvis det for to elementer $f(x,y)$ og $g(x,y)$ gælder, at vi ikke kan ordne dem efter ordningen siger vi, at $f(x,y)$ og $g(x,y)$ ikke er sammenlignelige. Det vil altså sige, at to polynomier ikke er sammenlignelige, hvis de har samme grad, og får deres grad fra den samme variabel.
Det ses at $\prec$ ikke er en total ordning, da hverken $x^{q^5} + x^{q^2} \prec x^{q^5} + x^{q^1}$ eller $x^{q^5} + x^{q^1} \prec x^{q^5} + x^{q^2}$. Det er dog en total ordning på monomier og det er derfor muligt at definere det ledende monomium $\mathrm{LM}_{\prec}(f)$ til at være det største monomium i f med hensyn til $\prec$.

Lemma 3.15:
Betragt to bivariate lineariserede polynomier $f(x,y)$ *og* $g(x,y)$*, som ikke er sammenlignelige. Der kan laves en linear kombination* $h(x,y) = f(x,y) + \gamma g(x,y)$ *således, at det for et passende* γ *gælder, at* $h(x,y) \prec f(x,y)$ *og* $h(x,y) \prec g(x,y)$

Bevis:
Antag, at vi har to bivariate lineariserede polynomier $f(x,y)$ og $g(x,y)$ som

ikke er sammenlignelige over $\prec$. Da f og g ikke er sammenlignelige må de have samme vægtede grad. Endvidere må det gælde, at $\mathrm{LM}_\prec(f) = \mathrm{LM}_\prec(g)$, for hvis vi ønsker, at kunne sammenligne polynomierne f og g med samme vægtet grad, siger ligningerne (3.5) og (3.6) at den variabel, der giver den højeste vægt skal være forskellig, og hvis den er det kan vi sammenligne disse. Heraf altså, at to polynomier, der ikke er sammenlignelige skal have samme grad og det skal være den samme variabel, der ligger til grund for graden, hvilket betyder, at de ledende monomier må være ens.
Vi kan vælge $\gamma = -\frac{\mathrm{LC}_\prec(f)}{\mathrm{LC}_\prec(g)}$ således at når vi regner $h(x,y) = f(x,y) + \gamma g(x,y)$ går $\mathrm{LT}_\prec(f)$ ud med $\mathrm{LT}_\prec(g)$ således, at der nu gælder, at $\mathrm{LM}_\prec(h) \prec \mathrm{LM}_\prec(f) = \mathrm{LM}_\prec(g)$ og dermed er $h(x,y) \prec f(x,y)$ og $h(x,y) \prec g(x,y)$.

■

Lad os nu definere x- og y-minimalitet af polynomier.

Definition 3.16:
Lad A være en mængde af r lineært uafhængige punkter $(x_i, y_i) \in W$. Vi siger, at et ikke-nul polynomium $f(x,y)$ er x-minimalt med hensyn til A hvis $f(x,y)$ er et minimalt polynomium under $\prec$ således, at $LM_\prec(f) = x^{q^{d_x(f)}}$ og $f(x,y) = 0$ for alle punkterne i A.

På samme måde kan et y-minimalt polynomium defineres.
Vi vil nu vise, at polynomierne, som genereres i algoritme 5 er minimale.

Sætning 3.17:
Polynomierne $f_0(x,y)$ og $f_1(x,y)$, der genereres i algoritme 5 er hhv. x- og y-minimalt med hensyn til den givne mængde af r lineært uafhængige punkter $(x_i, y_i) \in W$

Bevis:
Vi starter med at bemærke, at det altid er muligt at sammenligne x-minimale og y-minimale polynomier under $\prec$. Vi vil bevise sætningen ved brug af induktion.
Vi starter med basisskridtet, hvor vi skal vise, at $f_0(x,y) = x$ og $f_1(x,y) = y$ er hhv. x-minimalt og y-minimalt med hensyn til den tomme mængde. Dette er klart, da det ikke er muligt at finde et polynomium $h(x,y) \prec f_0(x,y)$ hvor

$\mathrm{LM}_{\prec}(h) = x^{q^{d_x(h)}}$. Det samme gør sig gældende for $f_1(x,y)$. Og alle polynomier evaluerer til nul over den tomme mængde.

I induktionsskridtet antager vi nu, at der for de første j skridt gælder, at $f_0(x,y)$ og $f_1(x,y)$ er hhv. x-minimalt og y-minimalt med hensyn til (x_i, y_i), hvor $i = 1, \ldots, j$.
Vi vil nu først vise, at de polynomier, der genereres i næste skridt $(j+1)$, vil evaluere til 0. Vi skal således vise, at følgende polynomier evaluerer til nul for alle (x_i, y_i) hvor $i = 1, \ldots, (j+1)$

$$\begin{aligned} f_1'(x,y) &= \Delta_1 f_0(x,y) - \Delta_0 f_1(x,y) \\ f_0'(x,y) &= f_0^q(x,y) - \Delta_0^{q-1} f_0(x,y) \\ f_0''(x,y) &= \Delta_1 f_0(x,y) - \Delta_0 f_1(x,y) \\ f_1''(x,y) &= f_1^q(x,y) - \Delta_1^{q-1} f_1(x,y). \end{aligned}$$

Det ses, at alle 4 evaluerer til nul for alle (x_i, y_i) hvor $i = 1, \ldots, j$. Vi mangler nu kun at vise, at det også gælder for (x_{j+1}, y_{j+1}). Vi har følgende - husk, at i hvert skridt er $\Delta_i = f_i(x_{j+1}, y_{j+1})$.

$$\begin{aligned} f_1'(x_{j+1}, y_{j+1}) &= \Delta_1 f_0(x_{j+1}, y_{j+1}) - \Delta_0 f_1(x_{j+1}, y_{j+1}) \\ &= f_1(x_{j+1}, y_{j+1}) f_0(x_{j+1}, y_{j+1}) - f_0(x_{j+1}, y_{j+1}) f_1(x_{j+1}, y_{j+1}) \\ &= 0. \end{aligned}$$

På samme måde kan $f_0''(x_{j+1}, y_{j+1})$ regnes. Vi har ydermere, at

$$\begin{aligned} f_0'(x_{j+1}, y_{j+1}) &= f_0^q(x_{j+1}, y_{j+1}) - \Delta_0^{q-1} f_0(x_{j+1}, y_{j+1}) \\ &= f_0^q(x_{j+1}, y_{j+1}) - f_0^{q-1}(x_{j+1}, y_{j+1}) f_0(x_{j+1}, y_{j+1}) \\ &= f_0^q(x_{j+1}, y_{j+1}) - f_0^q(x_{j+1}, y_{j+1}) \\ &= 0, \end{aligned}$$

og på samme måde kan $f_1''(x_{j+1}, y_{j+1})$ regnes. Vi har nu vist, at i skridt $j+1$ vil de konstruerede polynomier evaluere til nul for alle (x_i, y_i) hvor $i = 1, \ldots, (j+1)$.

Vi skal nu vise, at de polynomier, der konstrueres i skridt $j+1$ også er hhv. x- og y-minimalt. Dette gøres ved at undersøge de forskellige tilfælde, der er i algoritmen. Vi starter med at undersøge tilfældet hvor $\Delta_0 \neq 0$, $\Delta_1 \neq 0$ og $f_1(x,y) \prec f_0(x,y)$. Vi har da, at

$$f_0'(x,y) = \Delta_1 f_0(x,y) - \Delta_0 f_1(x,y).$$

I dette tilfælde har vi, at $\mathrm{LM}_{\prec}(f_0') = \mathrm{LM}_{\prec}(f_0)$ og dermed følger x-minimaliteten af $f_0'(x,y)$ af, at $f_0(x,y)$ er x-minimalt.
Vi har ligeledes, at

$$f_1'(x,y) = f_1^q(x,y) - \Delta_1^{q-1} f_1(x,y).$$

Vi skal vise, at $f_1'(x,y)$ er y-minimalt med hensyn til punkterne (x_i, y_i) hvor $i = 1, \ldots, (j+1)$. Dette gøres ved modstrid og vi antager derfor, at $f_1'(x,y)$ ikke er y-minimalt. Dette medfører, at der findes et y-minimalt polynomium $f_1''(x,y)$ med hensyn til punkterne (x_i, y_i) hvor $i = 1, \ldots, (j+1)$. Dette polynomium $f_1''(x,y)$ har samme ledende monomium som $f_1(x,y)$, da der må gælde, at $f_1''(x,y) \prec f_1'(x,y)$ så altså $d_y(f_1'') < d_y(f_1') = d_y(f_1) + 1$. Da $f_1(x,y)$ er y-minimalt kan $f_1''(x,y)$ ikke kan have mindre grad end $f_1(x,y)$ og derfor har de samme ledende monomium. Samtidig gælder det, at $f_1''(x,y) \neq f_1(x,y)$, hvilket kommer af, at $f_1''(x_{j+1}, y_{j+1}) = 0$, mens $f_1(x_{j+1}, y_{j+1}) \neq 0$. Vi ved nu fra lemma 3.15, at det er muligt at finde et $h(x,y)$ som er en lineær kombination af $f_1(x,y)$ og $f_1''(x,y)$, således, at $h(x,y) \prec f_1(x,y)$ og vi ved derfor også, at $h \prec f_0(x,y)$. Det gælder ydermere for h, at det evaluere til 0 for alle (x_i, y_i) hvor $i = 1, \ldots, j$ hvilket er i modstrid med y-minimaliteten af $f_1(x,y)$ eller x-minimaliteten af $f_0(x,y)$. Derfor må antagelsen om at $f_1'(x,y)$ ikke er y-minimalt være forkert. Et tilsvarende bevis kunne føres for $f_0(x,y) \prec f_1(x,y)$.

Vi undersøger nu det tilfælde hvor $\Delta_0 = 0$ og $\Delta_1 \neq 0$. I dette tilfælde er $f_0(x,y)$ uændret og er derfor stadig x-minimalt. Vi skal derfor blot undersøge om

$$f_1'(x,y) = f_1^q(x,y) - \Delta_1^{q-1} f_1(x,y)$$

er y-minimalt med hensyn til punkterne (x_i, y_i) hvor $i = 1, \ldots, (j+1)$. Hvis vi antager, at $f_1'(x,y)$ ikke er y-minimalt, er det igen muligt at finde et $f_1''(x,y)$ hvor $f_1''(x,y) \neq f_1(x,y)$, og hvor $f_1''(x,y)$ og $f_1(x,y)$ har samme ledende monomium. Disse er igen ikke ens da $f_1''(x_{j+1}, y_{j+1}) = 0$ og $f_1(x_{j+1}, y_{j+1}) \neq 0$. Det er nu muligt at lave en passende lineær kombination $h(x,y)$ af $f_1''(x,y)$ og $f_1(x,y)$ hvor der gælder at $h(x,y) \prec f_1(x,y)$. Hvis $\mathrm{LM}_{\prec}(h(x,y))$ er på formen $y^{q^{d_y(h)}}$ eller $h(x,y) \prec f_0(x,y)$ vil det være i modstrid med y-minimaliteten af $f_1(x,y)$ eller x-minimaliteten af $f_0(x,y)$.
Antag derfor, at dette ikke er tilfældet og dermed, at $LM_{\prec}(h)$ er på formen $x^{q^{d_x(h)}}$ og $h(x,y) \not\prec f_0(x,y)$. Det ses så, at $h(x_{j+1}, y_{j+1}) \neq 0$ da

$$h(x_{j+1}, y_{j+1}) = f_1''(x_{j+1}, y_{j+1}) + \gamma f_1(x_{j+1}, y_{j+1}) = 0 + \gamma f_1(x_{j+1}, y_{j+1}),$$

hvor $f_1(x_{j+1}, y_{j+1}) \neq 0$ og da $f_0(x_{j+1}, y_{j+1}) = 0$ kan $h(x, y)$ derfor ikke være et multiplum under $\otimes$ af $f_0(x, y)$.
Da h er på formen $x^{q^{d_x(h)}}$ og $f_0(x, y) \prec h(x, y)$ kan vi finde et t således, at $\mathrm{LM}_\prec(h(x, y)) = \mathrm{LM}_\prec(x^{q^t} \otimes f_0(x, y))$. Vi kan nu finde et polynomium $h''(x, y)$ som er en lineær kombination af $h(x, y)$ og $x^{q^t} \otimes f_0(x, y)$, og hvorom der gælder at $h''(x, y) \prec h(x, y)$. Hvis der for dette $h''(x, y)$ gælder, at enten $\mathrm{LM}_\prec(h'')$ er på formen $y^{q^{d_y(h'')}}$ eller $h''(x, y) \prec f_0(x, y)$ vil det være i modstrid med y-minimaliteten af $f_1(x, y)$ eller x-minimaliteten af $f_0(x, y)$. Hvis dette ikke er tilfældet laver vi på samme måde et nyt $h^{(3)}(x, y) \prec h''(x, y)$ dette kan vi forsætte med til vi finder et polynomium $\hat{h}(x, y)$ som enten har $\mathrm{LM}_\prec(\hat{h}(x, y)) = y^{q^{d_y(\hat{h})}}$ eller $\hat{h}(x, y) \prec f_0(x, y)$, som enten er i modstrid med y-minimaliteten af $f_1(x, y)$ eller x-minimaliteten af $f_0(x, y)$.
Tilfældet hvor $\Delta_0 \neq 0$ og $\Delta_1 = 0$ kan vises på samme måde, og i tilfældet hvor $\Delta_0 = 0$ og $\Delta_1 = 0$ er der ikke noget at vise.

■

Vi har således vist, at polynomierne $f_0(x, y)$ og $f_1(x, y)$ lavet i algoritme 5 opfylder kravet om at være bivariate lineariserede polynomier som evaluere til nul for alle (x_i, y_i) hvor $i = 1, \ldots, r$. De polynomier vi vælger mellem at returnere er hhv. x- og y-minimalt og vi returnerer det af dem, som har lavest grad. Vi har dermed opfyldt kravene til $Q(x, y)$ polynomiet.

Hvis der er sendt et l-dimensionelt rum $V \in \mathcal{K}$ og der er modtaget et $(l-\rho+t)$-dimensionelt underrum U af W foregår dekodningen på følgende måde.

- Kør algoritme 5 på en basis for U for at finde et bivariat lineariseret polynomium $Q(x, y) = Q_x(x) + Q_y(y)$ af minimal $(1, k-1)$-vægtet grad som evaluerer til nul over hele rummet U. Hvis et sådant polynomium ikke har både en $Q_x(x)$ og en $Q_y(y)$ del, returner "Fejl".

- Kør algoritme 4 med $(-Q_x(x), Q_y(x))$ som input for at finde et lineariseret polynomium $f(x)$ som har den egenskab, at $-Q_x(x) \equiv Q_y(x) \otimes f(x)$. Hvis et sådan polynomium ikke kan findes returner da "Fejl".

- Returner polynomiet $f(x)$ som informationspolynomiet, der korresponderer til kodeordet $\hat{V} \in \mathcal{K}$, hvis $d(U, \hat{V}) < l - k + 1$.

Tidskompleksiteten af denne procedure er domineret af algoritme 5 som har en kompleksitet på $O((l+m)^2)$ i $\mathbb{F}_{q^m}$ [12].

Lad os nu bevæge os videre og se et eksempel på dekodning.

Eksempel 3.18:
Lad os prøve at dekode de koder, som vi genererede i eksempel 3.11. Vi benytter den samme oversættelse mellem potenser af variablen s som i tabel 3.1. Indkodningen genererede disse vektorer.

a	10100000
b	00110010
c	01000101
d	11111010

I henhold til teorien skulle vi være i stand til at rette op til en enkelt fejl eller sletning. Lad os antage, at outputtet fra netværket er de følgende vektorer, som er lineært uafhængige kombinationer af indkodningsvektorerne og som udspænder vektorrummet U. I transmissionen er der blevet introduceret en enkelt bitfejl, som er markeret.

a+b	10010010
c	01000101
a+d	01011010
b+d	1100100$\underline{1}$
a+c	11100101

Vi har således modtaget er rum af dimension 5, hvor de 4 af dimensionerne er de samme som i det sendte rum, der er derfor ingen sletninger og $\rho = 0$. Til gengæld er der en fejlvektor og derfor er $t = 1$.
Vi opdeler vektorerne i følgende vektorpar.

x_1	1001	$= 1 + s^3$	$= s^{14}$	y_1	0010		$= s^2$
x_2	0100		$= s$	y_2	0101	$= s + s^3$	$= s^9$
x_3	0101	$= s + s^3$	$= s^9$	y_3	1010	$= 1 + s^2$	$= s^8$
x_4	1100	$= 1 + s$	$= s^4$	y_4	1001	$= 1 + s^3$	$= s^{14}$
x_5	1110	$= 1 + s + s^2$	$= s^{10}$	y_5	0101	$= s + s^3$	$= s^9$

Vi er nu klar til at begynde at interpolere for at finde det lineariserede bivariate polynomium $Q(x, y) = Q_x(x) + Q_y(y)$ af minimal $(1, k-1)$-vægtet grad.
Vi starter med at sætte polynomierne f_0 og f_1 til følgende værdier.

$$f_0(x, y) = x$$
$$f_1(x, y) = y$$

Vi interpolerer med vektorparrene et efter et. Først (x_1, y_1)

$$\begin{aligned}\Delta_0 &= f_0(x_1, y_1)\\ &= s^{14}\\ \Delta_1 &= f_1(x_1, y_1)\\ &= s^2\end{aligned}$$

Da hverken Δ_0 eller Δ_1 er nul, beregner vi den $(1, k-1)$-vægtede grad af hvert af polynomierne, som er givet ved formlen

$$deg_{1,k-1}(f(x,y)) := \max\{d_x(f), k-1+d_y(f)\}.$$

Så

$$\begin{aligned}deg_{1,k-1}(f_0) &= deg_{1,k-1}\left(x^{2^0}\right)\\ &= \max\{0, 3-1+(-\infty)\}\\ &= 0\end{aligned}$$

og

$$\begin{aligned}deg_{1,k-1}(f_1) &= deg_{1,k-1}\left(y^{2^0}\right)\\ &= \max\{(-\infty), 3-1+0\}\\ &= 2.\end{aligned}$$

Vi fortsætter i det tilfælde, hvor $deg_{1,k-1}(f_0) \leq deg_{1,k-1}(f_1)$.

$$\begin{aligned}f_0(x,y) &= f_0^2(x,y) + \Delta_0 f_0(x,y)\\ &= x^2 + s^{14}x\\ f_1(x,y) &= \Delta_1 f_0(x,y) + \Delta_0 f_1(x,y)\\ &= s^2x + s^{14}y\end{aligned}$$

Vi interpolerer for det næste vektorpar (x_2, y_2).

$$\begin{aligned}
\Delta_0 &= f_0(x_2, y_2) \\
&= s^2 + s^{14}s \\
&= s^2 + 1 \\
&= s^8 \\
\Delta_1 &= f_1(x_2, y_2) \\
&= s^2 s + s^{14}s^9 \\
&= s^3 + s^8 \\
&= s^{13}
\end{aligned}$$

Da hverken Δ_0 eller Δ_1 er nul beregner vi igen den $(1, k-1)$-vægtede grad af hvert af polynomierne.
Så

$$deg_{1,k-1}(f_0) = 1$$

og

$$deg_{1,k-1}(f_1) = 2.$$

Vi fortsætter i det tilfælde hvor $deg_{1,k-1}(f_0) \leq deg_{1,k-1}(f_1)$.

$$\begin{aligned}
f_0(x, y) &= (x^2 + s^{14}x)^2 + s^8(x^2 + s^{14}x) \\
&= x^4 + s^{13}x^2 + s^8x^2 + s^7x \\
&= x^4 + s^3x^2 + s^7x \\
f_1(x, y) &= s^{13}(x^2 + s^{14}x) + s^8(s^2x + s^{14}y) \\
&= s^{13}x^2 + s^{12}x + s^{10}x + s^7y \\
&= s^{13}x^2 + s^3x + s^7y
\end{aligned}$$

Vi interpolerer for det næste vektorpar (x_3, y_3).

$$\begin{aligned}
\Delta_0 &= f_0(x_3, y_3) \\
&= (s^9)^4 + s^3(s^9)^2 + s^7 s^9 \\
&= s^6 + s^6 + s \\
&= s \\
\Delta_1 &= f_1(x_3, y_3) \\
&= s^{13}(s^9)^2 + s^3 s^9 + s^7 s^8 \\
&= s + s^1 2 + 1 \\
&= s^6
\end{aligned}$$

Da hverken Δ_0 eller Δ_1 er nul beregner vi igen den $(1, k-1)$-vægtede grad af begge polynomier.
Så

$$deg_{1,k-1}(f_0) = 2$$

og

$$deg_{1,k-1}(f_1) = 2.$$

Vi fortsætter i tilfældet hvor $deg_{1,k-1}(f_0) \leq deg_{1,k-1}(f_1)$.

$$\begin{aligned}
f_0(x, y) &= (x^4 + s^3x^2 + s^7x)^2 + s(x^4 + s^3x^2 + s^7x) \\
&= x^8 + s^6x^4 + s^{14}x^2 + sx^4 + s^4x^2 + s^8x \\
&= x^8 + s^{11}x^4 + s^9x^2 + s^8x \\
f_1(x, y) &= s^6(x^4 + s^3x^2 + s^7x) + s(s^{13}x^2 + s^3x + s^7y) \\
&= s^6x^4 + s^9x^2 + s^{13}x + s^{14}x^2 + s^4x + s^8y \\
&= s^6x^4 + s^4x^2 + s^{11}x + s^8y
\end{aligned}$$

Vi interpolerer for det næste vektorpar (x_4, y_4).

$$\begin{aligned}
\Delta_0 &= f_0(x_4, y_4) \\
&= (s^4)^8 + s^{11}(s^4)^4 + s^9(s^4)^2 + s^8 s^4 \\
&= s^2 + s^{12} + s^2 + s^{12} \\
&= 0 \\
\Delta_1 &= f_1(x_4, y_4) \\
&= s^6(s^4)^4 + s^4(s^4)^2 + s^{11}s^4 + s^8 s^{14} \\
&= s^7 + s^{12} + 1 + s^7 \\
&= s^{11}
\end{aligned}$$

Da Δ_0 er nul opdaterer vi kun f_1.

$$\begin{aligned}
f_0(x, y) &= x^8 + s^{11}x^4 + s^9x^2 + s^8x \\
f_1(x, y) &= (s^6x^4 + s^4x^2 + s^{11}x + s^8y)^2 + s^{11}(s^6x^4 + s^4x^2 + s^{11}x + s^8y) \\
&= s^{12}x^8 + s^8x^4 + s^7x^2 + sy^2 + s^2x^4 + x^2 + s^7x + s^4y \\
&= s^{12}x^8 + x^4 + s^9x^2 + s^7x + sy^2 + s^4y
\end{aligned}$$

Vi interpolerer for det næste vektorpar (x_5, y_5).

$$\begin{aligned}
\Delta_0 &= f_0(x_5, y_5) \\
&= (s^{10})^8 + s^{11}(s^{10})^4 + s^9(s^{10})^2 + s^8 s^{10} \\
&= s^5 + s^6 + s^14 + s^3 \\
&= s^7 \\
\Delta_1 &= s^{12}(s^{10})^8 + (s^{10})^4 + s^9(s^{10})^2 + s^7 s^{10} + s(s^9)^2 + s^4 s^9 \\
&= s^2 + s^{10} + s^{14} + s^2 + s^4 + s^{13} \\
&= 0
\end{aligned}$$

Da Δ_1 er nul opdaterer vi kun f_0.

$$\begin{aligned}
f_0(x, y) &= (x^8 + s^{11}x^4 + s^9x^2 + s^8x)^2 + s^7(x^8 + s^{11}x^4 + s^9x^2 + s^8x) \\
&= x^{16} + s^7x^8 + s^3x^4 + sx^2 + s^7x^8 + s^3x^4 + sx^2 + x \\
&= x^{16} + x \\
f_1(x, y) &= s^{12}x^8 + x^4 + s^9x^2 + s^7x + sy^2 + s^4y
\end{aligned}$$

Vi har nu interpoleret med alle vores vektorpar og beregner nu den endelige $(1, k-1)$-vægtede grad af de to polynomier.
Så

$$deg_{1,k-1}(f_0) = 4$$

og

$$deg_{1,k-1}(f_1) = 3.$$

Da $deg_{1,k-1}(f_1) < deg_{1,k-1}(f_0)$ skal vi returnere $f_1(x, y)$, så

$$Q(x, y) = s^{12}x^8 + x^4 + s^9x^2 + s^7x + sy^2 + s^4y$$

Vi ønsker nu at finde det lineariserede polynomium $f(x)$ i ligningen

$$Q_y(x) \otimes f(x) + Q_x(x) \equiv 0.$$

Dette kan gøres ved at beregne **RDiv**$(-Q_x(x)$,$Q_y(x))$, hvilket i dette tilfælde er **RDiv**$(s^{12}x^8 + x^4 + s^9x^2 + s^7x$,$sx^2 + s^4x)$.
Vi initialiserer algoritmen med de to polynomier, kvotienten og resten.

$$\begin{aligned} a(x) &= s^{12}x^8 + x^4 + s^9x^2 + s^7x \\ b(x) &= sx^2 + s^4x \\ q(x) &= 0 \quad r(x) = 0; \end{aligned}$$

Vi beregner graden af polynomierne.

$$e = \deg(b(x)) = 1 \quad d = \deg(a(x)) = 3.$$

Da graden af $a(x)$ er større end graden af $b(x)$ $(d \geq e)$ udfører vi divisionen.

$$\begin{aligned} a_d = LC(a(x)) = s^{12} \quad b_e = LC(b(x)) = s \\ t(x) = \left(\frac{s^{12}}{s}\right)^8 x^4 = s^{13}x^4 \end{aligned}$$

Vi opdaterer a polynomiet og kvotienten

$$\begin{aligned} q(x) &= q(x) + t(x) = s^{13}x^4 \\ a(x) &= a(x) - b(x) \otimes t(x) \\ &= (s^{12}x^8 + x^4 + s^9x^2 + s^7x) + (s(s^{13}x^4)^2 + s^4(s^{13}x^4)) \\ &= s^{12}x^8 + x^4 + s^9x^2 + s^7x + s^{12}s^8 + s^2x^4 \\ &= s^8x^4 + s^9x^2 + s^7x \end{aligned}$$

Vi beregner graden af det nye $a(x)$.

$$d = 2.$$

Da vi stadigvæk har $d \geq e$ udfører vi divisionen igen.

$$a_d = s^8 \quad b_e = s$$
$$t(x) = \left(\frac{s^8}{s}\right)^8 x^2 = s^{11}x^2$$

Vi opdaterer $a(x)$ og kvotienten

$$\begin{aligned} q(x) &= s^{13}x^4 + s^{11}x^2 \\ a(x) &= (s^8x^4 + s^9x^2 + s^7x) + (s(s^{11}x^2)^2 + s^4(s^{11}x^2)) \\ &= s^8x^4 + s^9x^2 + s^7x + s^8x^4 + x^2 \\ &= s^7x^2 + s^7x \end{aligned}$$

Vi beregner graden af $a(x)$.

$$d = 1.$$

Da stadig $d \geq e$ udfører vi divisionen.

$$a_d = s^7 \quad b_e = s$$
$$t(x) = \left(\frac{s^7}{s}\right)^8 x = s^3x^2$$

Vi opdaterer a polynomiet og kvotienten

$$\begin{aligned} q(x) &= s^{13}x^4 + s^{11}x^2 + s^3x^2 \\ a(x) &= (s^7x^2 + s^7x) + (s(s^3x^2)^2 + s^4(s^3x^2)) \\ &= s^7x^2 + s^7x + s^7x^4 + s^7x^2 \\ &= 0 \end{aligned}$$

Vi beregner graden af $a(x)$.

$$d = 0.$$

Da graden af $a(x)$ nu er strengt mindre end graden af $b(x)$ kan vi ikke dividere. Vi sætter $r(x) = a(x)$ og returnerer kvotienten og resten.

$$(q(x), r(x)) = \left(s^{13}x^4 + s^{11}x^2 + s^3x^2, 0\right)$$

Proceduren returnerer polynomiet

$$s^{13}x^4 + s^{11}x^2 + s^3x^2 = (1,0,1,1)x^4 + (0,1,1,1)x^2 + (0,0,0,1)x,$$

hvilket afslutter dekodningen. Koefficienterne er beskedvektorerne og vi kan se, at koefficienterne $(0,0,0,1), (0,1,1,1), (1,0,1,1)$ er de samme, som vi valgte da vi lavede indkodningen. Vi har derfor dekodet beskeden korrekt og samtidig rettet en fejl.

Der er tre mulige steder vi kan indse, at dekodningen er forkert. Det første sted er efter interpolationen. Hvis vi får et polynomium ud, hvor y ikke indgår i nogen led, kan vi ikke dekode. Det andet sted er efter divisionen. Hvis der er en rest er dekodningen forkert. Til sidst kan vi se, at hvis antallet af koefficienter i vores endelige polynomium ikke svarer til det antal beskeder vi forventer, så er dekodningen forkert.

Lad os betragte et eksempel, hvor vi ser at hvis en vektor er fejlbehæftet er det lige meget hvor mange bit der er fejl i - det tæller stadig kun som en fejl.

Eksempel 3.19:
Lad os endnu en gang forsøge at dekode koderne genereret i eksempel 3.11. Husk, at vi kan rette en fejl eller sletning.
Indkodningen genererer disse vektorer.

a	10100000
b	00110010
c	01000101
d	11111010

Lad os antage, at vi modtager følgende vektorer. Der er i transmissionen sket tre bitfejl i den tredje vektor, hvilket vi markerer.

a+b	10010010
c	01000101
a+d	01011010
b+d	10001101
a+c	11100101

Interpolationen giver polynomiet $s^{12}x^8 + s^8x^4 + s^7x^2 + sy^2$, hvilket dividerer til beskederne $(1,0,1,1),(0,1,1,1),(0,0,0,1)$. Denne dekodning er korrekt.
Vi ser altså, at ligegyldigt hvor mange bitfejl, der er sket i den samme vektor tæller det kun som en fejl.

Eksemplet viser altså, at flere bitfejl i samme vektor kun tæller som en fejl. Dette er logisk, hvis vi tænker på, at en bitfejl i en vektor svarer til at miste den oprindelige vektor og få en ny med fejl. Det betyder altså, at hvis der sker en bitfejl i en vektor, som allerede er fejlbehæftet svarer dette til at miste en fejlvektor og få en ny fejlvektor, dette gør altså ingen forskel.
Lad os nu betragte et andet eksempel, der vil vise os hvilke implikationer en enkelt bitfejl kan have på vores mulighed for at dekode.

Eksempel 3.20:
Lad os igen prøve at dekode koden, som vi genererede i eksempel 3.11. Husk at vi i følge teorien kan rette en fejl eller sletning.
Indkodningen genererer disse vektorer.

a	10100000
b	00110010
c	01000101
d	11111010

Lad os antage, at vi har fået følgende vektorer ud af netværket - disse er lineære kombinationer af vektorerne fra indkodningen. Der er ikke introduceret hverken fejl eller sletninger.

a+b	10010010
c	01000101
a+d	01011010
a+c	11100101

Interpolationen returnerer polynomiet $s^6x^4+s^4x^2+s^{11}x+s^8y$, hvilket divideres

til beskederne $(1,0,1,1),(0,1,1,1),(0,0,0,1)$. Denne dekodning er tydeligvis korrekt.

Lad os nu antage, at vi modtager følgende tre vektorer, hvilket er de samme som før, men den tredje er forsvundet. Dette er en sletning.

a+b	10010010
c	01000101
a+c	11100101

Interpolationen returnerer polynomiet $s^{10}x^4+s^8x^2+x+s^{12}y$, hvilket dividerer til beskeden $(1,0,1,1),(0,1,1,1),(0,0,0,1)$. Dette er tydeligvis korrekt dekodet.

Lad os nu antage, at vi modtager følgende fire vektorer. Det er de samme som de første fire vi betragtede, men der er sket en fejl i den tredje.

a+b	10010010
c	01000101
a+d	0101101$\underline{1}$
a+c	11100101

Interpolationen returnerer polynomiet $x^8+s^{10}x^4+sx^2+s^{10}x+y$, hvilket dividerer til beskederne $(1,0,0,0),(1,1,1,0),(0,1,0,0),(1,1,1,0)$. Denne dekodning er helt tydeligt forkert - vi forventer kun tre beskedvektorer.
Selvom vi kun introducerede en enkelt fejl kan vi altså stadig ikke dekode. Dette sker fordi denne fejl repræsenterer både en fejl, men også en sletning. Når der sker en fejl svarer det til, at vi mister en rigtig vektor og får en ny forkert vektor. Det er ikke altid sådan, at det at miste en vektor medfører en sletning, idet vi kan have andre vektorer, som udspænder den samme dimension, som den vektor vi har mistet. I dette tilfælde er der dog ikke andre vektorer, som udspænder samme dimension, hvorfor det at miste denne vektor medfører en sletning. Vi har altså både en fejl og en sletning og kan ikke dekode. Dog ser vi af det foregående, at hvis bare vi kunne få slettet den tredje vektor ville vi kun have en sletning, og derfor være i stand til at dekode. Det kan derfor være muligt, at vælge delmængder af $k=3$ vektorer, prøve at dekode disse og se om vi på den måde kunne løse dekodningsproblemet.

En enkelt bitfejl kan altså have stor indflydelse på muligheden for at dekode. Det er dog meget vigtigt at bemærke, at selvom en bitfejl sker tidligt i netværket og derfor kan sprede sig til andre vektorer, så giver dette ikke flere fejl i tilfældet hvor vi bruger Kötter-Kschischang koder. Dette skyldes, at efter der er sket en fejl udspænder vi et nyt (forkert) underrum. Det betyder, at hvis vi laver linearkombinationer af vektorerne vil vi fortsat udspænde dette vektorrum, og altså ikke introducere flere fejl. Dermed er der altså ingen mulighed for fejlpropagering i netværket, hvilket er meget vigtigt i netværk, hvor vektorerne kombineres meget.
Som vi så i eksempel 3.20 kan en enkelt bitfejl medføre, at der sker både en sletning og en fejl. For at undgå, at dette er tilfældet kan man lave en ekstra indkodning inden vektorerne sendes ud på de forskellige kanter. Vi starter derfor med inden vi sender, at indkode vektorerne med en lineær kode. Modtageren skal nu blot starte med at dekode de enkelte vektorer, der modtages. Da det er en lineær kode, der benyttes, kan man dekode selvom de modtagende vektorer er linearkombinationer af de sendte vektorer. Hvis der nu sker fejl undervejs i netværket vil de lineære koder kunne rette de enkelte bitfejl, der er sket, og underrumskoderne vil så være i stand til at tage sig af de sletninger der muligvis er sket. Hvis en vektor ikke fejlkorrigeres korrekt i henhold til den lineære kode, altså hvis der er sket for mange fejl, kan vi stadig risikere at den genererer fejl i Kötter-Kschischang koden. Hvis man derfor ikke kan dekode Kötter-Kschischang koden vil det være gavnligt, at betragte de vektorer, som blev fejlkorrigeret i henhold til den lineære kode som sletninger, evt. en eller flere af gangen, og se bort fra den/dem når underrumskoden skal dekodes. Geil, Foshammer og Neve-Græsbøll har skrevet om dette i en artikel [6].
Lad os betragte et eksempel.

Eksempel 3.21:
Vi betragter igen koderne, som blev genereret i eksempel 3.11 og vi kan derfor rette en fejl eller sletning.
Indkodningen med Kötter-Kschischang koder genererer disse vektorer.

a	10100000
b	00110010
c	01000101
d	11111010

Lad os nu indkode disse vektorer med Hammingkoder. For en gennemgang af Hammingkoder se Justesen og Høholdts bog om emnet [11].

Da vores beskeder er på 8 tegn vælger vi en (12,8)-kode. Vi konstruerer vores matricer således, at vi kan rette en fejl. Lad os lave matricen D, så

$$D = \begin{bmatrix} 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 \\ 0 & 1 & 1 & 1 \\ 1 & 0 & 0 & 1 \\ 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 1 \\ 1 & 1 & 1 & 1 \end{bmatrix}$$

Vi har da generatormatricen

$$G = (I, D) = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 \end{bmatrix}$$

og paritetstjekmatricen er

$$H = (D^T, I) = \begin{bmatrix} 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 \\ 1 & 0 & 1 & 1 & 0 & 0 & 1 & 1 & 0 & 1 & 0 & 0 \\ 1 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 0 & 1 & 0 \\ 0 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 1 \end{bmatrix}$$

Vi kan nu indkode vores vektorer således, at

a	$(10100000)\mathrm{G} = 101000000011$
b	$(00110010)\mathrm{G} = 001100101111$
c	$(01000101)\mathrm{G} = 010001010111$
d	$(11111010)\mathrm{G} = 111110100011$

Vi antager nu, at vi har lavet de samme linearkombinationer, som i eksempel 3.20, og at vi har fået den samme fejl. Vi har altså følgende vektorer ud af netværket.

a+b	100100101100
c	010001010111
a+d	010110110000
a+c	111001010100

Vi skal nu foretage syndromdekodning på disse vektorer for at rette eventuelle fejl. Der eksisterer $2^{12-8} = 16$ sideklasser og vores syndromer er $12 - 8 = 4$ bit lange. Sideklasselederen er det ord med mindst vægt, som kan give det opstillede syndrom, når sideklasselederen sættes ind som r i Hr^T.
Vi har følgende sammenhæng mellem syndromer og sideklasseledere.

syndrom $= Hr^T$	Sideklasseleder (fejlfunktion)
$(0000)^T$	(000000000000)
$(0110)^T$	(100000000000)
$(0011)^T$	(010000000000)
$(0101)^T$	(001000000000)
$(0111)^T$	(000100000000)
$(1001)^T$	(000010000000)
$(1011)^T$	(000001000000)
$(1101)^T$	(000000100000)
$(1111)^T$	(000000010000)
$(1000)^T$	(000000001000)
$(0100)^T$	(000000000100)
$(0010)^T$	(000000000010)
$(0001)^T$	(000000000001)
$(1010)^T$	(010010000000)
$(1100)^T$	(010000010000)
$(1110)^T$	(100000001000)

Alle sideklasselederne med et 1-tal er entydige, svarende til den ene fejl vi kan rette. For hver modtagne vektor skal det gælde, at hvis vi finder dens syndrom, så skal det være $(0000)^T$ ellers er der sket fejl i denne vektor. Hvis der er sket fejl finder vi sideklasselederen for det syndrom vi får og trækker denne fra vektoren for at få den rettede vektor.
Lad os nu fejlkorrigere vores vektorer.

a+b	$H(100100101100)^T = (0000)^T$
c	$H(010001010111)^T = (0000)^T$
a+d	$H(010110\underline{1}0000)^T = (1111)^T$
a+c	$H(111001010100)^T = (0000)^T$

Vi kan se, at den tredje vektor er fejlbehæftet, som forventet. Vi korrigerer derfor i henhold til syndromet, så

$$010110\underline{1}0000 - 000000010000 = 010110100000,$$

hvilket giver $H(010110\underline{0}0000)^T = (0000)^T$. Vi har nu korrigeret fejlen, og som det ses er vi nu tilbage ved vektorer uden fejl, hvilket betyder, at vi kan dekode med Kötter-Kschischang, som i eksempel 3.20.

Vi forlader her Kötter-Kschischang koderne. Som vi har set er disse meget robuste overfor at beskeder forsvinder i netværket, men de er til gengæld forholdsvis sårbare overfor bitfejl i enkelte beskeder. Bitfejl kan ikke sprede sig til andre beskeder og på den måde gøre mere skade, men de kan til gengæld medføre sletninger, hvilket gør det svært at dekode.

Kapitel 4

Lineær netværkskodning i generelle netværkskodningsproblemer

Vi vil nu bevæge os fra multicast netværkskodningsproblemer videre til generelle netværkskodningsproblemer, men først vil vi benytte de værktøjer, som vi fik i kapitel 2, til at lave to sætninger, der endeligt knytter løseligheden af et multicast netværkskodningsproblem til værdien af transferpolynomiet.

Sætning 4.1:
Et multicast netværkskodningsproblem er løseligt hvis og kun hvis det tilhørende transferpolynomium er ikke-nul. Hvis multicast netværkskodningsproblemet er løseligt er det garanteret at lineære koder er tilstrækkelige for alle legemer $\mathbb{F}_q$*, hvor* $q > |\mathcal{R}|$.

Bevis:
For at et multicast netværksproblem kan være løseligt må der eksistere et flow af størrelse h til alle modtagere ifølge sætning 1.3 side 7. Vi har da "kun hvis"-delen fra sætning 1.10 side 15. For at vise "hvis"-delen antager vi, tranferpolynomiet er ikke-nul. Ifølge sætning 1.10 eksisterer der så flows af størrelse h til alle modtagere. Da der er ikke er nogle variable, der kan forekomme med en grad højere end $|\mathcal{R}|$, ved vi fra korollar 2.45 side 43, at der eksisterer en ikke-nul løsning hvis størrelsen af legemet er større end $|\mathcal{R}|$.

■

Sætning 4.1 kan uden videre sammensættes med sætning 1.10 og give følgende sætning.

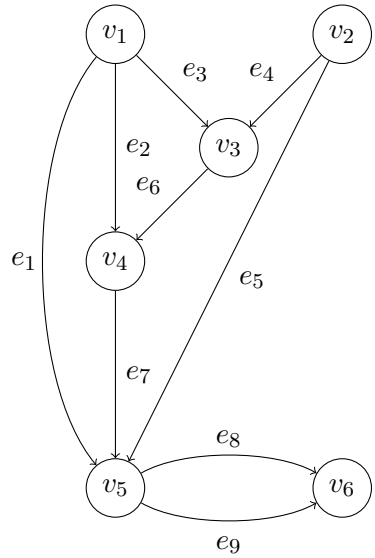
Sætning 4.2:
Et multicast netværkskodningsproblem er løseligt hvis og kun hvis der eksisterer et flow af størrelse h til alle modtagerne.

Lad os nu endeligt forlade multicast netværkskodningsproblemer og bevæge os videre til generelle netværkskodningsproblemer. Vi vil forsøge, at nå frem til en sætning, som tilsvarer sætning 4.1 for generel netværkskodning.
Vi betragter nu et generelt netværkskodningsproblem, med netværk $\mathcal{G}$ og en tilfældig mængde forbindelser $\mathcal{C}$. Vi fokuserer på lineær netværkskodning, hvilket betyder, at vi kan lave præcise udtalelser omkring et antal netværkskodningsproblemer.
For, at mængden af forbindelser (og dermed netværket) kan være løseligt skal vi sikre følgende to ting, for det første skal der være stort nok flow til alle forbindelser og for det andet må der ikke være forstyrrende interferens fra andre forbindelser.
Vi ønsker i høj grad at benytte den samme teori, som vi gjorde i afsnit 1.2. Den store forskel er bare, at i modsætning til multicast netværkskodning
Da vi ikke ønsker, at modtage alle beskeder i alle modtagere ønsker vi at redefinere $B^{(r)}$-matricen således, at vi ikke behøver regne på de dele af transfermatricen, som derfor ikke interesserer os. Vi definerer nu således, at for $r \in \mathcal{R}$ lad $B^{(r)}$ være en $|E| \times h$-matrix således, at

$$B_{i,j}^{(r)} = \begin{cases} b_{i,j}^{(r)} & \text{hvis } i \in ind(r) \text{ og } X_j \in \mathcal{D}(r) \\ 0 & \text{ellers.} \end{cases}$$

Dette sikrer, at vi kun får de dele af transfermatricen, som er relevante. Det eneste nye i definitionen er kravet om $X_j \in \mathcal{D}(r)$. Dette skyldes, at der nu ikke længere er tale om et multicast netværkskodningsproblem. Den samme definition kunne være brugt ved multicast netværkskodningsproblemer, idet det nye krav i den situation ikke ville have gjort nogen forskel.
Lad os betragte et eksempel, der kan illustrere brugen af transfermatricen til at afgøre om et netværkskodningsproblem er løseligt.

Eksempel 4.3:
Lad et netværk $\mathcal{G}$ være givet som på figur 4.1.



Figur 4.1: *Netværk $\mathcal{G}$*

Vi forestiller os, at vi vil have følgende forbindelser $\mathcal{C} = \{(v_1, v_4, \{X_1^{(v_1)}, X_2^{(v_1)}\}), (v_2, v_6, \{X_1^{(v_2)}, X_2^{(v_2)}\})\}$.
Vi konstruerer nu matricerne A og F på samme måde, som vi ville have gjort hvis vi havde skullet løse et multicast netværkskodningsproblem og matricerne $B^{(r)}$, for alle $r \in \mathcal{R}$, efter vores nye definition. A er i dette tilfælde en 4×9-matrix, så

$$A = \begin{bmatrix} a_{11} & a_{1,2} & a_{1,3} & 0 & 0 & 0 & 0 & 0 & 0 \\ a_{21} & a_{2,2} & a_{2,3} & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & a_{3,4} & a_{3,5} & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & a_{4,4} & a_{4,5} & 0 & 0 & 0 & 0 \end{bmatrix}$$

F er en 9×9-matrix, så

$$F = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & f_{1,8} & f_{1,9} \\ 0 & 0 & 0 & 0 & 0 & 0 & f_{2,7} & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & f_{3,6} & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & f_{4,6} & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & f_{5,8} & f_{5,9} \\ 0 & 0 & 0 & 0 & 0 & 0 & f_{6,7} & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & f_{7,8} & f_{7,9} \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

$B^{(v_4)}$ og $B^{(v_6)}$ er 9×4-matricer, så

$$B^{(v_4)} = \begin{bmatrix} 0 & 0 & 0 & 0 \\ b_{2,1} & b_{2,2} & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ b_{6,1} & b_{6,2} & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \quad B^{(v_6)} = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & b_{8,3} & b_{8,4} \\ 0 & 0 & b_{9,3} & b_{9,4} \end{bmatrix}$$

Vi er nu klar til at beregne de to transfermatricer $M^{(v_4)} = A(I - F)^{-1}B^{(v_4)}$ og $M^{(v_6)} = A(I - F)^{-1}B^{(v_6)}$, som er 4×4-matricer.

$$M^{(v_4)} = \begin{bmatrix} a_{1,2}b_{2,1} + a_{1,3}f_{3,6}b_{6,1} & a_{1,2}b_{2,2} + a_{1,3}f_{3,6}b_{6,2} & 0 & 0 \\ a_{2,2}b_{2,1} + a_{2,3}f_{3,6}b_{6,1} & a_{2,2}b_{2,2} + a_{2,3}f_{3,6}b_{6,2} & 0 & 0 \\ a_{3,4}f_{4,6}b_{6,1} & a_{3,4}f_{4,6}b_{6,2} & 0 & 0 \\ a_{4,4}f_{4,6}b_{6,1} & a_{4,4}f_{4,6}b_{6,2} & 0 & 0 \end{bmatrix}$$

$$M^{(v_6)} = \begin{bmatrix} 0 & 0 & M_{1,3}^{(v_6)} & M_{1,4}^{(v_6)} \\ 0 & 0 & M_{2,3}^{(v_6)} & M_{2,4}^{(v_6)} \\ 0 & 0 & M_{3,3}^{(v_6)} & M_{3,4}^{(v_6)} \\ 0 & 0 & M_{4,3}^{(v_6)} & M_{4,4}^{(v_6)} \end{bmatrix},$$

hvor indgangene i $M^{(v_6)}$ er givet ved

$$\begin{aligned}
M_{1,3}^{(v_6)} &= (a_{1,1}f_{1,8} + a_{1,2}f_{2,7}f_{7,8} + a_{1,3}f_{3,6}f_{6,7}f_{7,8})\, b_{8,3} + \\
&\quad (a_{1,1}f_{1,9} + a_{1,2}f_{2,7}f_{7,9} + a_{1,3}f_{3,6}f_{6,7}f_{7,9})\, b_{9,3} \\
M_{1,4}^{(v_6)} &= (a_{1,1}f_{1,8} + a_{1,2}f_{2,7}f_{7,8} + a_{1,3}f_{3,6}f_{6,7}f_{7,8})\, b_{8,4} + \\
&\quad (a_{1,1}f_{1,9} + a_{1,2}f_{2,7}f_{7,9} + a_{1,3}f_{3,6}f_{6,7}f_{7,9})\, b_{9,4} \\
M_{2,3}^{(v_6)} &= (a_{2,1}f_{1,8} + a_{2,2}f_{2,7}f_{7,8} + a_{2,3}f_{3,6}f_{6,7}f_{7,8})\, b_{8,3} + \\
&\quad (a_{2,1}f_{1,9} + a_{2,2}f_{2,7}f_{7,9} + a_{2,3}f_{3,6}f_{6,7}f_{7,9})\, b_{9,3} \\
M_{2,4}^{(v_6)} &= (a_{2,1}f_{1,8} + a_{2,2}f_{2,7}f_{7,8} + a_{2,3}f_{3,6}f_{6,7}f_{7,8})\, b_{8,4} + \\
&\quad (a_{2,1}f_{1,9} + a_{2,2}f_{2,7}f_{7,9} + a_{2,3}f_{3,6}f_{6,7}f_{7,9})\, b_{9,4} \\
M_{3,3}^{(v_6)} &= (a_{3,4}f_{4,6}f_{6,7}f_{7,8} + a_{3,5}f_{5,8})\, b_{8,3} + (a_{3,4}f_{4,6}f_{6,7}f_{7,9} + a_{3,5}f_{5,9})\, b_{9,3} \\
M_{3,4}^{(v_6)} &= (a_{3,4}f_{4,6}f_{6,7}f_{7,8} + a_{3,5}f_{5,8})\, b_{8,4} + (a_{3,4}f_{4,6}f_{6,7}f_{7,9} + a_{3,5}f_{5,9})\, b_{9,4} \\
M_{4,3}^{(v_6)} &= (a_{4,4}f_{4,6}f_{6,7}f_{7,8} + a_{4,5}f_{5,8})\, b_{8,3} + (a_{4,4}f_{4,6}f_{6,7}f_{7,9} + a_{4,5}f_{5,9})\, b_{9,3} \\
M_{4,4}^{(v_6)} &= (a_{4,4}f_{4,6}f_{6,7}f_{7,8} + a_{4,5}f_{5,8})\, b_{8,4} + (a_{4,4}f_{4,6}f_{6,7}f_{7,9} + a_{4,5}f_{5,9})\, b_{9,4}.
\end{aligned}$$

Transfermatricerne er $h \times h$-matricer, hvor h er antallet af beskeder og det gælder, at indgang $M_{i,j}^{(r)}$ består af alle mulige veje fra afsenderen af X_i til modtageren r, som ønsker X_j. Det betyder, at det ikke for alle indgange er den samme besked, der afsendes og ønskes modtaget, hvilket betyder, at det ikke er hele transfermatricen, det umiddelbart er relevant at fortolke algebraisk på.
Som tidligere ønsker vi, at determinanten af den matrix, som svarer til de beskeder, der ønskes modtaget, skal være ikke-nul. Dette betød i multicast, at vi ønskede at tage determinanten af hele transfermatricen, men nu betyder det, at vi skal udtage en delmatrix af transfermatricen og betragte dennes determinant.
Da vi har ordnet beskederne ens i A og $B^{(r)}$ matricerne betyder dette, at de også er ordnet ens i både rækker og søjler af transfermatricen. Vi ønsker derfor at udtage en delmatrix, som defineres af de beskeder, som modtageren ønsker, hvilket er det samme for rækker og søjler. Da vi har valgt vores $B^{(r)}$ matrix smart betyder dette samtidig, at denne delmatrix defineres af hvilke søjler i transfermatricen, som har ikke-nul indgange og de tilsvarende rækker.
Det betyder, at for modtager v_4 udtager vi matricen

$$\begin{bmatrix} M_{1,1}^{(v_4)} & M_{1,2}^{(v_4)} \\ M_{2,1}^{(v_4)} & M_{2,2}^{(v_4)} \end{bmatrix} = \begin{bmatrix} a_{1,2}b_{2,1} + a_{1,3}f_{3,6}b_{6,1} & a_{1,2}b_{2,2} + a_{1,3}f_{3,6}b_{6,2} \\ a_{2,2}b_{2,1} + a_{2,3}f_{3,6}b_{6,1} & a_{2,2}b_{2,2} + a_{2,3}f_{3,6}b_{62} \end{bmatrix}$$

og for modtager v_6 udtager vi matricen

$$\begin{bmatrix} M_{3,3}^{(v_6)} & M_{3,4}^{(v_6)} \\ M_{4,3}^{(v_6)} & M_{4,4}^{(v_6)} \end{bmatrix}$$

Vi ønsker nu, at determinanten af disse to matricer skal være ikke-nul, for at sikre, at de ønskede beskeder kommer frem. Dette betyder, at der er et stort nok flow mellem afsender og modtager, hvilket vi ifølge sætning 1.5 side 8 ved kræves for at netværket kan løses. Samtidig ønsker vi, at alle andre indgange i de to transfermatricer skal være 0, for at fjerne støj.
Vi får nu følgende ligningssystem

$$\begin{gathered} M_{1,1}^{(v_4)} \cdot M_{2,2}^{(v_4)} - M_{2,1}^{(v_4)} \cdot M_{1,2}^{(v_4)} \neq 0 \\ M_{3,3}^{(v_6)} \cdot M_{4,4}^{(v_6)} - M_{4,3}^{(v_6)} \cdot M_{3,4}^{(v_6)} \neq 0 \\ M_{3,1}^{(v_4)} = 0, \quad M_{3,2}^{(v_4)} = 0, \quad M_{4,1}^{(v_4)} = 0, \quad M_{4,2}^{(v_4)} = 0 \\ M_{1,3}^{(v_6)} = 0, \quad M_{1,4}^{(v_6)} = 0, \quad M_{2,3}^{(v_6)} = 0, \quad M_{2,4}^{(v_6)} = 0 \end{gathered}$$

Hvis der findes en løsning til dette ligningssystem er netværket løseligt. Ellers er netværket ikke løseligt.
I dette tilfælde er netværket ikke løseligt. Vi kan opfylde ligningerne $M_{3,1}^{(v_4)} = 0$, $M_{3,2}^{(v_4)} = 0$, $M_{4,1}^{(v_4)} = 0$ og $M_{4,2}^{(v_4)} = 0$ på tre forskellige måder. Enten er $f_{4,6} = 0$ eller $a_{3,4} = 0$ og $a_{4,4} = 0$ eller $b_{6,1} = 0$ og $b_{6,2} = 0$. I alle tre tilfælde får vi, at $M_{3,3}^{(v_6)} \cdot M_{4,4}^{(v_6)} - M_{4,3}^{(v_6)} \cdot M_{3,4}^{(v_6)} = 0$, hvilket betyder netværket ikke kan løses. Bemærk her, at dette ikke skyldes en mangel i lineær netværkskodning, netværkskodningsproblemet kan reelt ikke løses med nogen kodningsstrategi ifølge Kötter og Médard [13].

Vi ser altså, at vi ud fra transfermatricerne kan sige om et netværkskodningsproblem er løseligt eller ej. Dette kan samles til følgende sætning.

Sætning 4.4:
Lad et acyklisk lineært netværkskodningsproblem $(\mathcal{G}, \mathcal{C})$ være givet og lad for alle $r \in \mathcal{R}$ gælde, at $M^{(r)}$ er transfermatricen for modtager r. Netværket har en lineær løsning hvis og kun hvis der eksisterer en værditildeling til alle $a_{i,j}$, $f_{i,j}$ og $b_{i,j}$ således, at

1. *Hvis vi har $\mathcal{D}(r) = \{X_{i_1}, \ldots, X_{i_{\nu(r)}}\}$ skal matricen*

$$M_{\mathcal{D}}^{(r)} = \begin{bmatrix} M_{i_1,i_1}^{(r)} & M_{i_1,i_2}^{(r)} & \cdots & M_{i_1,i_{\nu(r)}}^{(r)} \\ M_{i_2,i_1}^{(r)} & \ddots & & \\ \vdots & & & \\ M_{i_{\nu(r)},i_1}^{(r)} & & & \end{bmatrix}$$

   være en ikke-singulær $\nu(r) \times \nu(r)$-matrix.

2. *$M_{i,j}^{(r)} = 0$ for alle par af beskeder (X_i, X_j), hvor $X_i \notin \mathcal{D}(r)$ eller $X_j \notin \mathcal{D}(r)$.*

Bevis:
Antag, at netværket har en værditildeling, som opfylder de to betingelser. Betingelse 2 sikrer, at der ikke er nogen interferens ved modtageren. Desuden kan enhver modtager ud fra matricen i betingelse 1 finde de ønskede beskeder, hvilket følger af sætning 4.1. Dette viser den ene vej.
Antag nu, at en af betingelserne ikke er opfyldt.
Hvis betingelse 1 ikke er opfyldt er der ikke nok flow til en given modtager og vi kan da ifølge sætning 1.5 ikke løse netværkskodningsproblemet.
Hvis betingelse 2 ikke er opfyldt er mængden af beskeder $\mathcal{Z}(r)$ i en modtager r en blanding af beskeder og interferens, idet vi har nogle uønskede beskeder på en eller flere kanter i $ind(r)$. Dette betyder, at modtager r ikke har mulighed for at skelne mellem de ønskede beskeder og interferens, og derfor ikke kan reproducere $\mathcal{D}(r)$ og dermed er netværket ikke løseligt.

■

Vi har altså nu præcise betingelser for hvornår et netværk er løseligt. Det er dog en langtrukken opgave at tjekke betingelserne, for det kræver at der skal findes frem til en værditildeling, som opfylder betingelserne. Vi betragter en algebraisk tilgang til dette og lader i det følgende alle variablerne være samlet i $\bar{\xi} = \{a_1, \ldots, a_i, f_1, \ldots, f_j, b_1, \ldots, b_k\}$. Lad $g_1(\bar{\xi}), g_2(\bar{\xi}), \ldots, g_K(\bar{\xi})$ være alle de

indgange i alle $M^{(r)}$, som skal være nul for at opfylde betingelse 2 i sætning 4.4. Vi betragter nu idealet, som genereres af $g_1(\bar{\xi}), g_2(\bar{\xi}), \ldots, g_K(\bar{\xi})$ og noterer dette ideal som $I(g_1, g_2, \ldots, g_K)$.

Sætning 4.5:
Idealet $I(g_1, g_2, \ldots, g_K)$ er et egentligt ideal for $\mathbb{F}_2[\bar{\xi}]$ hvis og kun hvis vi kan finde en værditildeling for $\bar{\xi}$, som opfylder betingelse 2 i sætning 4.4.

Bevis:
Vi starter med at vise "kun hvis"-delen og antager at vi ikke kan lave en værditildeling, der opfylder betingelse 2 i sætning 4.4. Det betyder at varieteten af idealet er tom, og ifølge Hilberts Nullstellensatz 2.39 side 41 er idealet dermed uegentligt.
Lad os nu vise "hvis"-delen og antager, at vi har en værditildeling, der opfylder betingelse 2 i sætning 4.4. Vi har dermed et nulpunkt og altså en varietet af idealet, som er ikke tom. Ifølge Hilberts Nullstellensatz 2.39 er idealet egentligt.

■

For at opfylde betingelse 1 lader vi $h_1(\bar{\xi}), h_2(\bar{\xi}), \ldots, h_L(\bar{\xi})$ være determinanterne af $M_{\mathcal{D}}^{(r)}$ matricerne. Disse skal være forskellige fra 0. Vi introducerer en variabel ξ_0 og betragter funktionen $1 - \xi_0 \prod_{i=1}^{L} h_i(\bar{\xi})$. Vi kalder idealet $I(g_1(\bar{\xi}), g_2(\bar{\xi}), \ldots, g_K(\bar{\xi}), 1-\xi_0 \prod_{i=1}^{L} h_i(\bar{\xi}))$ for idealet for det lineære netværkskodningsproblem og noterer det $Ideal((\mathcal{G},\mathcal{C}))$. Den algebraiske varietet, som er associeret med Ideal$((\mathcal{G},\mathcal{C}))$ noteres Var$((\mathcal{G},\mathcal{C}))$ og er givet ved

$$\text{Var}((\mathcal{G},\mathcal{C})) = \{(x_1, x_2, \ldots, x_n) \in \mathbb{F}^n \mid g(x_1, x_2, \ldots, x_n) = 0 \text{ for alle } g \in \text{Ideal}((\mathcal{G},\mathcal{C}))\}.$$

Vi relaterer nu løseligheden af et lineært netværkskodningsproblem til varietetens størrelse.

Sætning 4.6:
Lad et lineært netværkskodningsproblem $(\mathcal{G},\mathcal{C})$ være givet.
Netværkskodningsproblemet er løseligt hvis og kun hvis Var$((\mathcal{G},\mathcal{C}))$ er ikke-tom og dermed, at idealet Ideal$((\mathcal{G},\mathcal{C}))$ er et egentligt ideal af $\bar{\mathbb{F}}[\xi_0, \bar{\xi}]$, altså Ideal$((\mathcal{G},\mathcal{C})) \subset \mathbb{F}_2[\xi_0, \bar{\xi}]$.

Bevis:
Antag først at idealet Ideal($(\mathcal{G},\mathcal{C})$) er et egentligt ideal af $\mathbb{F}_2[\xi_0,\bar{\xi}]$. Hilberts Nullstellensatz 2.39 giver så, at varieteten af I er ikke nul. Altså $V(\text{Ideal}((\mathcal{G},\mathcal{C}))) \neq \emptyset$, hvilket betyder, at $\text{Var}((\mathcal{G},\mathcal{C})) \neq \emptyset$, da polynomierne i idealet har fælles nulpunkter. Det betyder, at vi kan finde en værditildeling til $\bar{\xi}$ og ξ_0 således, at betingelse 2 i sætning 4.4 er opfyldt. Ydermere vil vi for alle løsninger i varieteten $\text{Var}((\mathcal{G},\mathcal{C}))$ have, at $\xi_0 \neq 0$ og $\prod_{i=1}^{L} h_i(\bar{\xi}) \neq 0$, da vi ellers ville have 1 som generator i idealet, og derfor ville $\text{Ideal}((\mathcal{G},\mathcal{C})) = \mathbb{F}_2[\xi_0,\bar{\xi}]$, hvilket er i modstrid med antagelsen. Vi har dermed, at betingelse 1 i sætning 4.4 er opfyldt og ethvert element i $V(\text{Ideal}((\mathcal{G},\mathcal{C})))$ er en løsning til det lineære netværkskodningsproblem.
Omvendt antag nu, at $\text{Ideal}((\mathcal{G},\mathcal{C})) = \mathbb{F}_2[\xi_0,\bar{\xi}]$, altså idealet er uegentligt. Dette betyder ifølge sætning 4.5, at ingen værditildeling opfylder betingelse 2 i sætning 4.4.
Dette viser sætningen. Vi ser desuden, at hvis vi for en løsning til netværkskodningsproblemet vælger en passende værdi for ξ_0, så vil varieteten $\text{Var}((\mathcal{G},\mathcal{C}))$ altid være ikke-tom.

■

Ved brug af sætning 4.6 har vi reduceret spørgsmålet om et netværkskodningsproblems løselighed til at spørgsmål om at bestemme hvorvidt dets varietet er tom eller ej. Vi kan besvare dette spørgsmål ved at benytte Buchbergers algoritme 3 til at finde en Gröbnerbasis for idealet $\text{Ideal}((\mathcal{G},\mathcal{C}))$. Det er velkendt, at Gröbnerbasen for et ideal indeholder 1 hvis og kun hvis den tilsvarende varietet er tom.
Lad os nu omsætte teorien til et eksempel.

Eksempel 4.7:
Betragt igen netværkskodningsproblemet fra eksempel 4.3. Vi har altså netværket i figur 4.1 og forbindelserne $\mathcal{C} = \{(v_1, v_4, \{X_1^{v_1}, X_2^{v_1}\}), (v_2, v_6, \{X_1^{v_2}, X_2^{v_2}\})\}$. Fra eksempel 4.3 har vi ligningssystemet

$$M_{1,1}^{(v_4)} \cdot M_{2,2}^{(v_4)} - M_{2,1}^{(v_4)} \cdot M_{1,2}^{(v_4)} \neq 0$$
$$M_{3,3}^{(v_6)} \cdot M_{4,4}^{(v_6)} - M_{4,3}^{(v_6)} \cdot M_{3,4}^{(v_6)} \neq 0$$
$$M_{3,1}^{(v_4)} = 0, \quad M_{3,2}^{(v_4)} = 0, \quad M_{4,1}^{(v_4)} = 0, \quad M_{4,2}^{(v_4)} = 0$$
$$M_{1,3}^{(v_6)} = 0, \quad M_{1,4}^{(v_6)} = 0, \quad M_{2,3}^{(v_6)} = 0, \quad M_{2,4}^{(v_6)} = 0.$$

Dette betyder, at vi har

$$\begin{aligned}
Ideal((\mathcal{G},\mathcal{C})) &= I(g_1(\bar{\xi}), g_2(\bar{\xi}), \ldots, g_K(\bar{\xi}), 1 - \prod_{i=1}^{L} h_i(\bar{\xi})) \\
&= I(M_{3,1}^{(v_4)}, M_{3,2}^{(v_4)}, M_{4,1}^{(v_4)}, M_{4,2}^{(v_4)}, M_{1,3}^{(v_6)}, M_{1,4}^{(v_6)}, M_{2,3}^{(v_6)}, M_{2,4}^{(v_6)}, \\
&\quad 1 - \left(\left(M_{1,1}^{(v_4)} \cdot M_{2,2}^{(v_4)} - M_{2,1}^{(v_4)} \cdot M_{1,2}^{(v_4)}\right)\right. \\
&\quad \left.\cdot \left(M_{3,3}^{(v_6)} \cdot M_{4,4}^{(v_6)} - M_{4,3}^{(v_6)} \cdot M_{3,4}^{(v_6)}\right)\right)
\end{aligned}$$

Lad os nu køre Buchbergers algoritme, som er algoritme 3, på polynomierne i idealet. Pointen med Buchbergers algoritme er at udvide den basis, der udspændes af polynomierne til en Gröbnerbasis. Det betyder, at vi finder S-polynomierne af alle par af polynomier modulo mængden af polynomier, og tilføjer de fremkomne polynomier (med undtagelse af nulpolynomiet) til vores basis. Vi viser her et par enkelte eksempler på udregningerne. Bemærk, at G' er den nuværende basis.

$$\begin{aligned}
\overline{S(g_1(\bar{\xi}), g_2(\bar{\xi}))}^{G'} &= \overline{S(a_{3,4}f_{4,6}b_{6,1}, a_{3,4}f_{4,6}b_{6,2})}^{G'} \\
&= \overline{\frac{a_{3,4}f_{4,6}b_{6,1}b_{6,2}}{a_{3,4}f_{4,6}b_{6,1}} \cdot a_{3,4}f_{4,6}b_{6,1} - \frac{a_{3,4}f_{4,6}b_{6,1}b_{6,2}}{a_{3,4}f_{4,6}b_{6,2}} \cdot a_{3,4}f_{4,6}b_{6,2}}^{G'} \\
&= 0,
\end{aligned}$$

så her skal der ikke tilføjes noget til vores basis.

$$\overline{S(g_2(\bar{\xi}), 1 - \prod_{i=1}^{L} h_i(\bar{\xi}))}^{G'} = 1,$$

så vi tilføjer 1 til vores Gröbnerbasis.
Da vi nu ser, at 1 tilhører vores Gröbnerbasis ved vi, at varieteten for idealet af netværkskodningsproblemet er tom, hvilket ifølge sætning 4.6 betyder, at netværkskodningsproblemet ikke er løseligt, hvilket vi også så i eksempel 4.3.

Der er stadig mange åbne spørgsmål inden for lineær netværkskodning. Det kunne være interessant at undersøge hvordan successandsynlighederne er for, at et netværkskodningsproblemet bliver løst, hvis man vælger funktionerne

tilfældigt og om sandsynligheden vil vokse hvis alfabetstørrelsen vokser, på samme måde som det gør sig gældende for lineær multicast netværkskodningsproblemer. Ligeledes kunne det være interessant at undersøge om der er en topologisk fortolkning på mellemregningerne når man finder Gröbnerbasen, eller om Gröbnerbasen for et løselige netværkskodningsproblem fortæller noget om antallet af løsninger eller har en anden topologisk fortolkning. Disse spørgsmål er dog uden for dette speciales fokus, og vi forlader derfor her det lineære netværkskodningsproblem og vender os mod det generelle tilfælde.

Kapitel 5

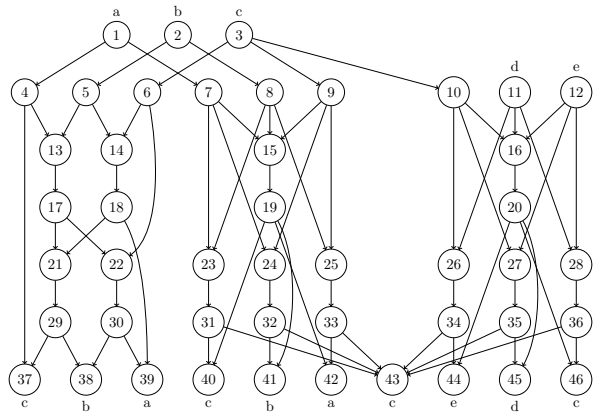
Generel netværkskodning

Vi ønsker at undersøge om generel netværkskodning har nogle fordele, som lineær netværkskodning ikke har.
Vi vil nu fokusere på det generelle netværkskodningsproblem, altså netværkskodningsproblemer, hvor hver enkelt modtager ikke nødvendigvis ønsker at modtage alle beskederne, men kun en delmængde af disse. Vi vil primært betragte netværkskodningsproblemer, hvor modtagerne kun ønsker at modtage en enkelt besked hver.
Dougherty et al. [4] præsenterer et eksempel på et netværkskodningsproblem, som ikke kan løses med lineær netværkskodning. Det er dog muligt at løse netværkskodningsproblemet ved hjælp af generel netværkskodning.
Vi vil i det følgende betragte dette eksempel både i sin helhed og i mindre dele.

Sætning 5.1:
Netværkskodningsproblemet illustreret på figur 5.1 har ingen lineær løsning.

Bevis:
Vi ved fra sætning 4.6, at der findes en lineær løsning hvis og kun hvis varieteten er ikke-tom. Det er muligt at opsætte matricerne A, F og $B^{(r)}$ for alle $r \in \mathcal{R}$ og vi kan nu på samme måde som i eksempel 4.3 lave matricerne $M^{(r)}$. På samme måde som i eksempel 4.7 finder vi en Gröbnerbasis for idealet for netværkskodningsproblemet. Vi finder, at vores Gröbnerbasis indeholder 1 og varieteten er derfor tom. Dermed findes der ikke en lineær løsning til netværkskodningsproblemet illustreret på figur 5.1.

■



Figur 5.1: *Et netværkskodningsproblem, som ikke kan løses med lineær netværkskodning, men som er løseligt.*

5.1 Brute force

Det første vi forsøgte var at afprøve alle kombinationer af funktioner i de enkelte indkodningspunkter og se hvilke af disse, der gav det ønskede resultat. I første omgang arbejdede vi med alfabetstørrelse 2, altså binært. Dette medfører, at hvis vi f.eks. kigger på punkt 13, som får 2 beskeder ind, er der 16 mulige funktioner, som det ses i tabel 5.1.

input \ funktion	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
(0,0)	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1
(0,1)	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1
(1,0)	0	0	0	0	1	1	1	1	0	0	0	0	1	1	1	1
(1,1)	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1

Tabel 5.1: *De mulige funktioner for et indkodningspunkt med to input og alfabetstørrelse 2.*

Tabel 5.1 skal læses på følgende måde. Hvis man skal bruge funktion 10 og får 0 som input på den første indgang og 1 på den anden, altså (0,1), så sender man 1 videre, og modtager man (1,0) sender man 0 videre. Bemærk, at søjlen under funktionsnummeret svarer til nummeret skrevet i binært. Man kan således ved at skrive funktionsnummeret binært finde ud af hvad denne funktion gør. Dette betyder, at vi ikke behøver at have denne tabel skrevet ud, men blot kan kigge på funktionsnummeret skrevet binært.

Normalt vil vi ikke betragte vores funktioner som tabeller, men derimod betragte dem som matricer af størrelse $|A| \times |A|$, hvis A er vores alfabet og der er to input. F.eks. vil matricen for funktion 10 se ud som følger

$$\begin{array}{c} \\ 0 \\ 1 \end{array} \begin{array}{c} \begin{array}{cc} 0 & 1 \end{array} \\ \begin{bmatrix} 0 & 1 \\ 0 & 1 \end{bmatrix} \end{array},$$

hvor rækkerne symboliserer en fast værdi af første input og søjlerne en fastholdt værdi af andet input.

Vi vil i det følgende eksempel, forklare hvordan vi ser om en funktionstildeling er en løsning.

Eksempel 5.2:
I figur 5.1 betragtes det venstre delnetværk, som består af punkterne $\{1, 2, 3, 4, 5, 6, 13, 14, 17, 18, 21, 22, 29, 30, 37, 38, 39\}$. For at konstruere en løsning til dette netværkskodningsproblem i $\mathbb{F}_2$ er der tilfældigt valgt funktioner til alle indkodningspunkter. Vi ønsker nu, at tjekke om de valgte funktioner løser netværkskodningsproblemet.
Når netværket initialiseres skal alle kombinationer af beskeder sendes gennem netværket, så modtagerne kan lave dekodningsfunktioner. Da netværkskodning
Vi begynder med at tjekke for punkt 37, som er en modtager, der ønsker at modtage beskeden c. Da punkt 4 kun har én kant ind, er dette ikke et indkodningspunkt, hvorfor funktionen på kanten $e_{4,37}$ blot er $f(x) = x$, altså en videresendelse af inputtet, som er beskeden a. Punkt 29 er heller ikke et indkodningspunkt, så på kant $e_{29,37}$ får vi også blot videresendt inputtet, som kommer fra punkt 21. Punkt 21 er et indkodningspunkt, og vi antager, at den valgte funktion giver output $(1, 0, 1, 0, 0, 0, 1, 1)$ på kanten $e_{21,29}$.
Vi har altså følgende i punktet 37.
Input: $(x_1, \ldots, x_8) = a = (0, 1, 0, 1, 0, 1, 0, 1)$ og $(y_1, \ldots, y_8) = (1, 0, 1, 0, 0, 0, 1, 1)$.
Ønsket output: $(v_1, \ldots, v_8) = c = (0, 0, 0, 0, 1, 1, 1, 1)$.
Det betyder, at $(x_1, y_1) = (0, 1)$ skal dekode til 0 og $(x_7, y_7) = (0, 1)$ skal dekode til 1. Vi har altså, at to forskellige (x, y) par, nemlig 1 og 7, skal dekode til forskellige v værdier. Dette kalder vi en falsk dublet, og vi kan derfor ikke konstruere en dekodningsfunktion i punkt 37, som kan dekode c entydigt ud fra disse input, da vi ikke ved om $(x, y) = (0, 1)$ skal dekodes til 0 eller 1. Den valgte kombination af funktioner er derfor ikke en løsning.

Vi vil nu definere en falsk dublet formelt.

Definition 5.3 (Falsk dublet):
For ordnede mængder $(x_1^{(1)}, \ldots, x_n^{(1)}), \ldots, (x_1^{(s)}, \ldots, x_n^{(s)}), (v_1, \ldots, v_n)$ *gælder det, at hvis der eksisterer værdier* $i \neq j$ *således, at* $x_i^{(k)} = x_j^{(k)}$ *for alle* k*, og* $v_i \neq v_j$*, kaldes dette for en falsk dublet.*

I eksempel 5.2 siger vi, at en falsk dublet medfører, at c ikke kan dekodes entydigt. Dette skyldes, at vi ikke kan lave en funktion således, at de samme input kan give to forskellige outputs. Hvis vi derimod ikke har nogen falske dubletter betyder det, at vi entydigt kan sige hvilket output vi skal få af alle forekommende input. Dette giver os følgende sætning.

Sætning 5.4:
Betragt en mulig løsning til et netværksproblem, hvor en modtager r *får de ordnede inputmængder* $(x_1^{(1)}, \ldots, x_n^{(1)}), \ldots, (x_1^{(s)}, \ldots, x_n^{(s)})$ *og ønsker, at kunne udlede de ordnede mængder* $(v_1^{(1)}, \ldots, v_n^{(1)}), \ldots, (v_1^{(t)}, \ldots, v_n^{(t)})$*. Hvis der ikke er nogle falske dubletter, når de ønskede mængder bruges som outputmængde en ad gangen, så er den mulige løsningen en løsning til netværkskodning*

5.2 Tilfældig netværkskodning

Når enten alfabetstørrelsen eller netværket vokser, bliver antallet af kombinationer af funktioner så stort, at det ikke er muligt at afprøve alle mulige kombinationer. Hvis vi f.eks. har alfabetstørrelse 4 og kun kigger på den del af netværket til venstre som i eksempel 5.2, har vi $\left(4{,}3 \cdot 10^9\right)^4 = 3{,}4 \cdot 10^{38}$ kombinationer, da vi har 4 punkter med to input som hver skal vælge én funktion ud af $4^{16} = 4{,}3 \cdot 10^9$, hvilket er så mange, at det ikke på nogen måde kan lade sig gøre teste alle disse kombinationer.
Vi er dog interesserede i at undersøge, om andelen af kombinationer, der løser netværkskodningsproblemet, er stort nok til, at det vil give mening at vælge funktionerne tilfældigt. Vi ved, at hvis vi betragter et multicast netværkskodningsproblem, er det for lineær netværkskodning tilstrækkeligt at lave sine funktioner over $\mathbb{F}_q$, hvor $q \geq |R|$ for at sikre, at sandsynligheden for at vælge en rigtig kombination er positiv og for q stort nok er sandsynligheden for at vælge en rigtig kombination af funktioner gående mod 1. Derfor giver det mening at

teste om dette også kan virke for generel netværkskodning og generelle netværkskodningsproblemer. Det giver mening at undersøge, om vi ved tilfældigt valg kan finde en løsning.

Eksempel 5.5 (Tilfældig netværkskodning i $\mathbb{F}_4$):
Vi ved fra Dougherty et al. [4], at det kan lade sig gøre at løse det venstre delnetværk af figur 5.1 i $\mathbb{F}_4$, løsningen fra artiklen er, at alle 4 indkodningspunkter bruger funktionen

$$\begin{array}{c} \\ 0 \\ 1 \\ 2 \\ 3 \end{array} \begin{array}{c} \begin{array}{cccc} 0 & 1 & 2 & 3 \end{array} \\ \begin{bmatrix} 0 & 1 & 2 & 3 \\ 1 & 0 & 3 & 2 \\ 2 & 3 & 0 & 1 \\ 3 & 2 & 1 & 0 \end{bmatrix} \end{array}$$

Vi valgte at forsøge ved tilfældigt valg af funktioner at finde en løsning til netværkskodningsproblemet i $\mathbb{F}_4$. Vi kørte 10.000.000 forsøg uden at finde en løsning og det viste sig ydermere, at vi ikke engang fandt en løsning, der løste problemet for den første modtager.
Da vi ved, at der findes mindst en løsning til netværkskodningsproblemet i $\mathbb{F}_4$, virker det som om tilgangen med tilfældigt valgte koder ikke er effektivt.

5.3 Begrænsning af antallet af mulige løsninger

Som vi så i eksempel 5.5 tyder det på, at det ikke er muligt at vælge indkodningsfunktionerne tilfældigt mellem alle mulige funktioner. Dette skyldes formodentligt, at sandsynligheden for at ramme en kombination af indkodningsfunktioner, der løser problemet, er forsvindende lille. Vi vil derfor nu forsøge at begrænse mængden af funktioner, der vælges imellem, ved at skære en række funktioner fra. På den måde vil vi søge, at øge sandsynligheden for tilfældigt at vælge en kombination af indkodningsfunktioner, som løser netværkskodningsproblemet.
Alle funktioner har tre muligheder for at håndtere input - enten glemmer de dem helt, delvist eller husker dem helt. Som vi senere skal se kan der være tidspunkter, hvor det er nødvendigt, at glemme et input fuldstændigt, men ellers virker det logisk, at huske på hele inputtet i stedet for kun at huske dele

af det.
Vi giver følgende definition af en funktion, der husker dele af sine input.

Definition 5.6 (Afhængighed):
En funktion $f(X_1, \ldots, X_k, Y)$ siges at afhænge af Y hvis der eksisterer værdier $x_1, \ldots, x_k, y_1, y_2$, hvor $y_1 \neq y_2$ således, at

$$f(x_1, \ldots, x_k, y_1) \neq f(x_1, \ldots, x_k, y_2)$$

Det betyder altså, at hvis funktionen er afhængig af et givet input, så husker den i hvert fald lidt om inputtet. Samtidig vil det sige, at hvis en funktion er uafhængig af et input vil funktionen glemme alt om dette input. Det betyder altså, at vi kan konstruere en funktion g, således, at $f(X, Y, Z) = g(X, Z)$, hvis f ikke afhænger af Y.
Vi kan nu definere en lineær funktion som en funktion på formen $f(X_1, \ldots, X_n) = a_0 + a_1X_1 + \cdots + a_nX_n$, som er afhængig af alle sine variabler, dvs. at a_1 til a_n ikke må være nul.
Vi giver følgende definition af en funktion, som husker alt om nogle af sine input.

Definition 5.7 (Reversibel):
Betragt en funktion $f(X_1, \ldots, X_k)$. Vi siger, at $f(X_1, \ldots, X_k)$ er reversibel i variablen X_i, hvis der eksisterer en funktion h, således, at

$$h(X_1, \ldots, X_{i-1}, X_{i+1}, \ldots, X_k, f(X_1, \ldots, X_k)) = X_i$$

En funktion kan være reversibel i flere variabler. Funktionen $f(X_1, \ldots, X_k)$ er reversibel i variablene $\{X_{i_1}, \ldots, X_{i_s}\}$ hvis der findes funktioner

$$\begin{aligned} h_1(X_1, \ldots, X_{i_1-1}, X_{i_1+1}, \ldots, X_k, f(X_1, \ldots, X_k)) &= X_{i_1} \\ &\vdots \\ h_s(X_1, \ldots, X_{i_s-1}, X_{i_s+1}, \ldots, X_k, f(X_1, \ldots, X_k)) &= X_{i_s} \end{aligned}$$

Vi vil i det følgende omtale en funktion, der er reversibel i alle sine variabler,

som en reversibel funktion. Hvis ikke funktionen er reversibel i alle variabler vil dette blive specificeret. En funktion, der er reversibel husker altså alt om de variabler den er reversibel i. Det vil sige, at en funktion er afhængig af de variabler den er reversibel i. Det betyder samtidig, at hvis $f(X, Y, Z)$ er reversibel i Z, så findes der en funktion h, så $h(X, Y, f(X, Y, Z)) = Z$. Da dette gælder for alle værdier af X, Y og Z kan vi finde en anden funktion h', hvor vi har valgt værdien x således, at $h'(Y, f(x, Y, Z)) = Z$. Dette kan gøres generelt for k variable.

Eksempel 5.8:
Vi vil nu kigge på en reversibel funktion $f(X, Y)$. Da der skal findes en funktion h der opfylder at $h(X, f(X, Y)) = Y$ må der gælde at hvis Y ændre værdi må værdien for $f(X, Y)$ også ændre sig. Dette skyldes, at hvis $f(X, y_1) = f(X, y_2)$, så er $y_1 = h(X, f(X, y_1)) = h(X, f(X, y_2)) = y_2$ altså er $y_1 = y_2$. Vi skal således lave en funktion hvor funktionsværdien ændre sig hvis den ene variabel ændres, dette svarer til at lave en $q \times q$-matrix, hvor hvert element fra alfabetet forekommer præcist en gang i hver række og søjle.
Hvis vi har alfabetet størrelse 4, kunne en reversibel funktion $f(X, Y)$ se således ud

$$\begin{bmatrix} a & b & c & d \\ b & d & a & c \\ d & c & b & a \\ c & a & d & b \end{bmatrix}$$

En sådan matrice kaldes for et latinsk kvadrat. For mere information om disse matricer og hvordan disse genereres se appendix A.

Det er svært, at sige meget generelt om reversible funktioner. Lad os derfor indføre følgende.

Definition 5.9 (Ligefordelt):
En funktion $f(X_1, \ldots, X_k)$ hvor $X_i \in \mathbb{F}_n$ er ligefordelt, hvis vi for et alfabet $A = \{a_1, \ldots, a_n\}$ har, at

$$|A_1| = \cdots = |A_n|,$$

$$\textit{hvor } A_i = \{(x_1, \ldots, x_k) \mid f(x_1, \ldots, x_k) = a_i\}$$

Det betyder altså, at alle symboler, som kan være resultatet af en ligefordelt funktion er det lige mange gange. Det er lettere at arbejde med funktioner, som er ligefordelte, men vi kan dog ikke være sikre på hvor meget de husker om deres input. Det viser sig dog, at der er følgende sammenhæng mellem reversible og ligefordelte funktioner.

Sætning 5.10:
Mængden af reversible funktioner er en ægte delmængde af mængden af ligefordelte funktioner.

Bevis:
Betragt en funktion $f(X_1, \ldots, X_k)$ i $\mathbb{F}_n$ og antag, at denne er reversibel. Hvis vi vælger værdier $x_1, \ldots, x_{k-1}$ så kan X_k antage n forskellige værdier. Da f er reversibel må den antage en ny værdi for alle n værdier af X_k. Hvis vi ændrer på værdien af et X_i, $i \in \{1, \ldots, k-1\}$ kan X_k igen antage n forskellige værdier, og f må antage ligeså mange. Vi kan altså se, at f antager alle værdier lige mange gange og at den derfor er ligefordelt. Dette viser reversible funktioner $\subseteq$ ligefordelte funktioner.
For at vise, at reversible funktioner ikke er lig med ligefordelte funktioner konstruerer vi nu en ligefordelt funktion, som ikke er reversibel. Vi betragter en funktion $f(X, Y) = Y + 2XY + 2X^2Y$, som er en ligefordelt funktion i $\mathbb{F}_3$, dette kan ses i følgende matrix, som indeholder lige mange af hvert symbol.

$$\begin{array}{c} \\ 0 \\ 1 \\ 2 \end{array} \begin{array}{c} \begin{array}{ccc} 0 & 1 & 2 \end{array} \\ \left[\begin{array}{ccc} 0 & 1 & 2 \\ 0 & 2 & 1 \\ 0 & 1 & 2 \end{array}\right] \end{array}$$

Vi kan se, at for alle inputs, hvor vi vælger værdien $y = 0$ vil $f(X, 0)$ give 0. Dermed kan vi ikke aflæse X i disse tilfælde og vi kan altså ikke konstruere en funktion h, så $h(0, f(X, 0)) = X$. Funktionen $f(X, Y)$ er altså ikke reversibel, og vi får reversible funktioner $\subset$ ligefordelte funktioner.

■

Vi kan altså se, at ved at skære mængden af funktioner, vi vælger vores løsninger imellem, ned til de ligefordelte funktioner får vi stadig alle funk-

tioner, der husker alt om deres input.
Vi ønsker nu, at retfærdiggøre, at denne begrænsning af funktioner vi vælger imellem er rimelig og har derfor følgende formodninger.

Formodning 5.11:
Hvis der til et netværkskodningsproblem findes en løsning, som består af en eller flere funktioner, som er ikke-reversible findes der mindst en løsning, der kun består af reversible funktioner.

Formodning 5.12:
For et løseligt netværkskodningsproblem er sandsynligheden for at vælge en løsning tilfældigt mellem alle mulige kombinationer af funktioner mindre end sandsynligheden for at vælge en løsning tilfældigt mellem alle kombinationer af reversible funktioner. Det betyder, at en forholdsvis større procentdel af de reversible funktioner kan løse netværket.

Som en yderligere understøttelse af rimeligheden i denne begrænsning påviser vi, at de lineære funktioner er indeholdt i de reversible.

Sætning 5.13:
En lineær funktion på formen $f(X_1, \ldots, X_n) = a_0 + \sum_{i=1}^{n} a_i X_i$, *hvor* $a_i \neq 0$ *og* $X_i \in \mathbb{F}_q$ *for alle* $i \geq 1$, *er reversibel.*

Bevis:
Vi ser, at hvis vi vælger værdier $x_1, \ldots, x_{k-1}, x_{k+1}, \ldots, x_n$, så vil værdien af f ændre sig for alle valg af X_k, Dette viser altså, at funktionen er reversibel.

■

Det er ikke i alle tilfælde at lineære funktioner er en ægte delmængde af reversible funktioner, som vi kan se af følgende sætning.

Sætning 5.14:
Mængden af reversible funktioner af to variabler er lig med mængden af de lineære funktioner af to variabler i $\mathbb{F}_3$.

Bevis:
Da vi ved, at de lineære funktioner er indeholdt i de reversible, er det nok, at vise, at der ikke er flere reversible funktioner end der er lineære, for at vise

at alle reversible funktioner i $\mathbb{F}_3$ er lineære. Først regner vi antallet af lineære funktioner. Disse er på formen $f(X,Y) = a + bX + cY$ hvor vi kan vælge a, b og c tilfældigt i alfabetet, dog skal lineære funktioner afhænge af alle sine variable, og derfor kan b og c ikke tage værdien 0, dette medfører at a kan tage 3 forskellige værdier og b og c kan hver tage 2 forskellige værdier hvilket giver et samlet antal af lineære funktioner på $3 \cdot 2 \cdot 2 = 12$.
Vi vil nu finde antallet af reversible funktioner i $\mathbb{F}_3$. Vi kan afbilde en funktion med to variabler i $\mathbb{F}_3$ som en 3×3-matrix hvor der for alle rækker og søjler skal gælde, at alle elementerne fra alfabetet skal forekomme præcis én gang.

$$\begin{array}{c} \\ 0 \\ 1 \\ 2 \end{array} \begin{array}{c} \begin{array}{ccc} 0 & 1 & 2 \end{array} \\ \begin{bmatrix} v_{0,0} & v_{0,1} & v_{0,2} \\ v_{1,0} & v_{1,1} & v_{1,2} \\ v_{2,0} & v_{2,1} & v_{2,2} \end{bmatrix} \end{array}$$

Når vi fylder denne matrice starter vi med at vælge $v_{0,0}$, som kan tage 3 forskellige værdier. Derefter vælger vi $v_{0,1}$ som kan tage 2 forskellige værdier, da den ikke må være det samme som $v_{0,0}$. Dette giver kun ét valg til $v_{0,2}$. Når vi nu skal vælge $v_{1,0}$ ved vi den ikke må være det samme som $v_{0,0}$ så den kan nu tage samme værdi som $v_{0,1}$ eller $v_{0,2}$.
Hvis $v_{1,0} = v_{0,1}$ skal $v_{1,1} = v_{0,2}$ da vi ellers ville have, at $v_{1,2} = v_{0,2}$ hvilket medfører at den sidste søjle nu har to ens elementer hvilket ikke må forekomme. Hvis $v_{1,0} = v_{0,2}$ skal $v_{1,1} = v_{0,0}$ ellers ville vi igen få to ens elementer i samme søjle. Det betyder altså, at når $v_{1,0}$ er valgt, er der kun et valg for $v_{1,1}$ og når første disse elementer er givet er der kun 1 valg tilbage på de resterende pladser i matricen. Dette giver os, at det samlede antal af reversible funktioner i $\mathbb{F}_3$ er $3 \cdot 2 \cdot 2 = 12$, som er det samme som antallet af lineære funktioner. Da de lineære funktioner er en delmængde af de reversible og der er lige mange af dem har vi altså, at alle reversible funktioner er lineære i $\mathbb{F}_3$.

■

Mængden af reversible funktioner indeholder dog mere end blot de lineære i andre legemer, hvilket vi viser med følgende eksempel.

Eksempel 5.15:
Vi ønsker at vise, at vi kan finde en reversibel funktion i $\mathbb{F}_4$, som ikke er lineær.

Hvis vi betragter input og output for denne matrix

$$\begin{array}{c} \\ 0 \\ 1 \\ \alpha \\ \alpha^2 \end{array} \begin{array}{c} \begin{array}{cccc} 0 & 1 & \alpha & \alpha^2 \end{array} \\ \begin{bmatrix} \alpha^2 & \alpha & 1 & 0 \\ 0 & \alpha^2 & \alpha & 1 \\ \alpha & 1 & 0 & \alpha^2 \\ 1 & 0 & \alpha^2 & \alpha \end{bmatrix} \end{array},$$

ses det, at den er reversibel, idet alle symboler optræder én gang i hver række og søjle. Funktionen som laver denne matrice kan dog ikke være lineær på formen $f(X,Y) = a + bX + cY$ da der må gælde

$$\begin{aligned} f(0,0) = a + b \cdot 0 + c \cdot 0 &= \alpha^2 \\ a &= \alpha^2 \end{aligned}$$

og

$$\begin{aligned} f(1,0) = \alpha^2 + b \cdot 1 + c \cdot 0 &= 0 \\ \alpha^2 + b &= 0 \\ b &= \alpha^2 \end{aligned}$$

og

$$\begin{aligned} f(0,1) = \alpha^2 + \alpha^2 \cdot 0 + c \cdot 1 &= \alpha \\ \alpha^2 + c &= \alpha \\ c &= \alpha + (\alpha + 1) \\ c &= 1 \end{aligned}$$

hvilket medfører at det følgende skal gælde

$$\begin{aligned} f(1,1) = \alpha^2 + \alpha^2 \cdot 1 + 1 \cdot 1 &= \alpha^2 \\ \alpha^2 + \alpha^2 + 1 &= \alpha^2 \\ 1 &= \alpha^2 \end{aligned}$$

Men da dette ikke er tilfældet kan den funktion der generer matricen ikke være lineær.
Vi kan altså se, at der findes reversible funktioner, som ikke er lineære.

Vi ønsker nu, at fremføre en række sætninger, som kan hjælpe os med yderligere at begrænse mængden af mulige funktioner til løsning af et netværk.
De første to sætninger handler om hvordan vi kan omskrive givet forskellige inputs.

Sætning 5.16:
For en funktion $f(X,Y,Z)$, som er reversibel i Y og Z, kan vi finde funktioner g og h således, at

$$h(X, f(X,Y,Z)) = g(Y,Z),$$

hvor g er reversibel i Y og Z.

Bevis:
Betragt funktionen $f(X,Y,Z)$, som er reversibel i Y og Z.
Da f er reversibel i Y, må det for $y_i \neq y_j$ gælde, at $f(x_k, y_i, z_l) \neq f(x_k, y_j, z_l)$ for alle k, l. Specielt, må det gælde, at $f(x_1, y_i, z_l) \neq f(x_1, y_j, z_l)$ for alle l.
Vi vælger nu $g(Y,Z) = f(x_1, Y, Z)$, der, som det ses af ovenstående overvejelser, er reversibel i Y. Samtidig kan vi vælge h, så $h(X, f(X,Y,Z)) = h_1(f(x_1,Y,Z)) = g(Y,Z)$.
Tilbage er nu at vise, at g også er reversibel i Z.
Da f er reversibel i Z har vi en funktion $h'(X,Y,f(X,Y,Z)) = Z$. Det betyder, at vi kan finde en funktion $h''(Y, f(x_1,Y,Z)) = Z$, hvilket betyder, at $h''(Y, f(x_1,Y,Z)) = h''(Y, g(Y,Z)) = Z$, så g er reversibel i Z.

■

Sætning 5.17:
For reversible funktioner $f(X,Y)$ og $g(Y,Z)$ kan vi finde funktioner h og k således, at

$$h(f(X,Y), g(Y,Z)) = k(X,Z),$$

hvor k er reversibel i X og Z.

Bevis:
Betragt funktioner $f(X,Y)$ og $g(Y,Z)$, som er reversible alle deres variabler.
Da f er reversibel i X har vi, at for $x_i \neq x_j$ vil $f(x_i, y_k) \neq f(x_j, y_k)$ for alle k. Specielt vil $f(x_i, y_1) \neq f(x_j, y_1)$. Vi vælger en funktion $k_1(X) = f(X, y_1)$, der, som det ses af ovenstående overvejelser, er reversibel i X.
På samme måde har vi, da $g(Y,Z)$ er reversibel i Z, at for $z_i \neq z_j$ vil

$g(y_k, z_i) \neq g(y_k, z_j)$ for alle k og specielt for $k = 1$. Vi vælger nu en funktion $k_2(Z) = g(y_1, Z)$, som er reversibel i Z.
Vi vælger funktionen $\bar{k}(k_1(X), k_2(Z))$ således, at den er reversibel i k_1 og k_2, hvilket vil sige, at den også er reversibel i X og Z, da vi ved, at $\bar{k}(k_1(x_1), k_2(z)) \neq \bar{k}(k_1(x_2), k_2(z))$ for $k_1(x_1) \neq k_1(x_2)$, hvilket kun kan ske, hvis $x_1 \neq x_2$ og ligedan for k_2 og Z.
Vi kan nu konstruere $k(X, Z) = \bar{k}(k_1(X), k_2(Z))$, som er reversibel i X og Z.
Betragt nu funktionen $h(f(X, Y), g(X, Y))$. For h findes der $k \times k = k^2$ mulige kombinationer af inputs, hvor $k = |A|$, når A er det alfabet vi arbejder over. De tre variabler X, Y og Z har k^3 mulige kombinationer, men hvis vi ser bort fra Y får vi k^2 mulige kombinationer af X og Z.
Da de to funktioner er reversible i Y har vi $f(x_k, y_i) \neq f(x_k, y_j)$ og $g(y_i, z_l) \neq g(y_j, z_l)$ for alle k, l, i, j, hvor $i \neq j$. Vi konstruerer h' således, at $h'(f(X, y_i), g(y_i, Z)) = h'(f(X, y_j), g(y_j, Z))$, for $i \neq j$ hvilket betyder, at h' kan afbilde en funktion af X og Z, da Y ikke betyder noget i sig selv. Vi kan derfor sætte $h' = \bar{k}$.
Vi har nu $h(f(X, Y), g(Y, Z)) = h'(f(X, y_1), g(y_1, Z)) = h'(k_1(X), k_2(Z)) = \bar{k}(k_1(X), k_2(Z)) = k(X, Z)$.

■

De næste sætninger er begrænsende sætninger.

Sætning 5.18:
For en funktion $f(X, Y, Z)$, har vi, at hvis der eksisterer en funktion h, så $h(X, f(X, Y, Z)) = Z$, så kan $f(X, Y, Z)$ ikke afhænge af Y.

Bevis:
Vi ønsker, at bevise dette ved modstrid, og antager, at f afhænger af Y.
Vi har derfor, at der eksisterer værdier x_1, y_1, y_2, z_1, så $y_1 \neq y_2$, hvorom det gælder, af $f(x_1, y_1, z_1) \neq f(x_1, y_2, z_1)$. Samtidig er

$$h(x_1, f(x_1, y_1, z_1)) = z_1 = h(x_1, f(x_1, y_2, z_1))$$

Vi kan uden tab af generalitet antage, at $f(x_1, y_1, z_1) = 0$ og $f(x_1, y_2, z_1) = 1$ og har derfor $h(x_1, 0) = h(x_1, 1) = z_1$.
For alle værdier $z_i \neq z_j$ har vi, at $h(x_1, f(x_1, y_1, z_i)) \neq h(x_1, f(x_1, y_1, z_j))$, hvilket kræver, at $f(x_1, y_1, z_i) \neq f(x_1, y_1, z_j)$, altså må f være reversibel i Z. Funktionen $f(x_1, y_1, z_i)$ kan antage k forskellige værdier, hvor $k = |A|$, hvis A er vores alfabet. Vi ved, at hvis $f(x_1, y_1, z_i) = 0$, så er $h(x_1, 0) = z_1$ og dermed

må vi have, at $i = 1$. På samme måde, hvis $f(x_1, y_1, z_i) = 1$, så er $h(x_1, 1) = z_1$ og derfor er $i = 1$.
Vi mangler nu at tildele værdier til $k-1$ $f(x_1, y_1, z_i)$'er, men vi har kun $k-2$ værdier tilbage at tildele dem. Dette giver modstriden og $f(X, Y, Z)$ kan derfor ikke afhænge af Y.

■

Sætning 5.19:
For en funktion $f(X, Y)$ gælder, at hvis der findes en funktion h således, at

$$h(f(X, Y), Z) = X,$$

så kan $f(X, Y)$ ikke afhænge af Y.

Bevis:
Betragt funktioner $f(X, Y)$ og $h(f(X, Y), Z) = X$.
Vi har $h(f(x_1, y_1), Z) = x_1$ for alle værdier af Z. Det vil altså sige, at lige meget hvordan vi ændrer på Z vil værdien af $h(f(x_1, y_1), Z)$ ikke ændre sig. Vi kan altså konkludere, at $h(f(X, Y), Z)$ må være uafhængig af Z, hvilket betyder, at vi har, at $h_1(f(X, Y)) = X$.
Vi viser nu ved modstrid, at $f(X, Y)$ ikke kan afhænge af Y og antager derfor, at $f(X, Y)$ er afhængig af Y.
Der eksisterer derfor værdier x_1, y_1 og y_2, $y_1 \neq y_2$ således, at $f(x_1, y_1) \neq f(x_1, y_2)$. Vi kan uden tab af generalitet antage, at $f(x_1, y_1) = 0$ og $f(x_1, y_2) = 1$. Vi har nu $h_1(f(x_1, y_1)) = h_1(f(x_1, y_2)) = x_1$, altså $h(0) = h(1) = x_1$.
Vi ved, at $f(X, Y)$ kan antage k forskellige værdier, hvor $k = |A|$, hvis A er alfabetet, og samtidig kan X antage k forskellige værdier. Det vil sige, at idet $h(f(x_i, y_1)) = x_i$ må $f(x_i, y_1)$ antage forskellige værdier for alle i, $f(X, Y)$ er altså reversibel i X. Men hvis $f(x_i, y_1) = 0$, så er $h(0) = x_1$, så vi må have $i = 1$. På samme måde hvis $f(x_i, y_1) = 1$, så er $h(1) = x_1$, så vi må have $i = 1$. Det betyder, at for de resterende $k-1$ værdier af x_i, har vi kun $k-2$ værdier at tildele $f(x_i, y_1)$, hvilket er i modstrid med at $f(x_i, y_1)$ skal antage forskellige værdier for alle x_i.
Vi ser derfor, at $f(X, Y)$ ikke kan være afhængig af Y.

■

Sætning 5.20:
For funktioner $f(X, Y)$ og $g(Y, Z)$, hvor g er reversibel i Z, gælder, at hvis der

findes en funktion h, så

$$h(f(X,Y),g(Y,Z)) = X,$$

må $f(X,Y)$ være uafhængig af Y.

Bevis:
Betragt funktioner $f(X,Y)$, $g(Y,Z)$ og $h(f(X,Y),g(Y,Z)) = X$, hvor g er reversibel i Z.
Da $g(Y,Z)$ er reversibel i Z har vi, at $g(y_k,z_i) \neq g(y_k,z_j)$ for alle k,i,j, hvor $i \neq j$. Vi har $h(f(x_1,y_1),g(y_1,Z)) = x_1$ for alle værdier af Z. Det vil altså sige, at lige meget hvordan vi ændrer på Z og dermed på værdien af $g(y_1,Z)$ vil værdien af $h(f(x_1,y_1),g(y_1,Z))$ ikke ændre sig. Vi kan altså konkludere, at $h(f(X,Y),g(Y,Z))$ må være uafhængig af $g(Y,Z)$, hvilket betyder, at vi har, at $h_1(f(X,Y)) = X$.
Nu kan beviset afsluttes på samme måde, som beviset for sætning 5.19.

■

Sætning 5.21:
For funktioner $f(X,Y)$ og $g(X,Y,Z)$, hvor g er reversibel i Z, gælder, at hvis der eksisterer en funktion h således, at

$$h(f(X,Y),g(X,Y,Z)) = X,$$

så kan $f(X,Y)$ ikke være afhængig af Y.

Bevis:
Beviset føres som beviset for sætning 5.20.

■

Sætning 5.22:
For funktioner $f(X,Y)$ og $g(X,Y)$, hvor f er reversibel i X og g er reversibel i X og Y gælder det, at hvis der findes en funktion h således, at

$$h(f(X,Y),g(X,Y)) = X,$$

så er $f(X,Y)$ uafhængig af Y.

Bevis:
Betragt funktioner $f(X,Y)$, $g(X,Y)$, hvor f er reversibel i X og g er reversibel i X og Y, og $h(f(X,Y),g(X,Y)) = X$. Beviset føres ved modstrid, og vi

antager derfor, at $f(X,Y)$ er afhængig af Y.
Da f er afhængig af Y eksisterer der værdier x_1, y_1, y_2, $y_1 \neq y_2$ således, at $f(x_1,y_1) \neq f(x_1,y_2)$. Vi kan uden tab af generalitet antage, at $f(x_1,y_1) = 0$ og $f(x_1,y_2) = 1$. Da g er reversibel i Y har vi, at $g(x_k,y_i) \neq g(x_k,y_j)$ for alle k,i,j, $i \neq j$. Det betyder specielt, at $g(x_1,y_1) \neq g(x_1,y_2)$ og vi kan uden tab af generelitet antage, at $g(x_1,y_1) = 0$ og $g(x_1,y_2) = 1$. Det betyder, at $h(f(x_1,y_1),g(x_1,y_1)) = h(0,0) = x_1$ og $h(f(x_1,y_2),g(x_1,y_2)) = h(1,1) = x_1$.
Funktionen $f(X,y_1)$ kan maksimalt antage k forskellige værdier når X varieres, hvor $k = |A|$, hvis A er alfabetet. Det samme gælder for $g(X,y_1)$. Det betyder, at der er k par $(f(X,y_1),g(X,y_1))$, da både f og g ændres, når X ændres.
Idet $h(f(X,y_1),g(X,y_1)) = X$ skal alle disse par være forskellige, for de forskellige værdier af X. Men idet $h(0,0) = h(1,1) = x_1$ har vi kun $k-2$ forskellige par til de sidste $k-1$ værdier af X, hvilket giver modstriden. Dermed kan $f(X,Y)$ altså ikke være afhængig af Y.

■

Vi ser altså, at vi kan begrænse mængden af brugbare funktioner ved at kigge på hvad vi skal kunne udlede efterfølgende.
Som et sidste redskab indføres følgende sætning.

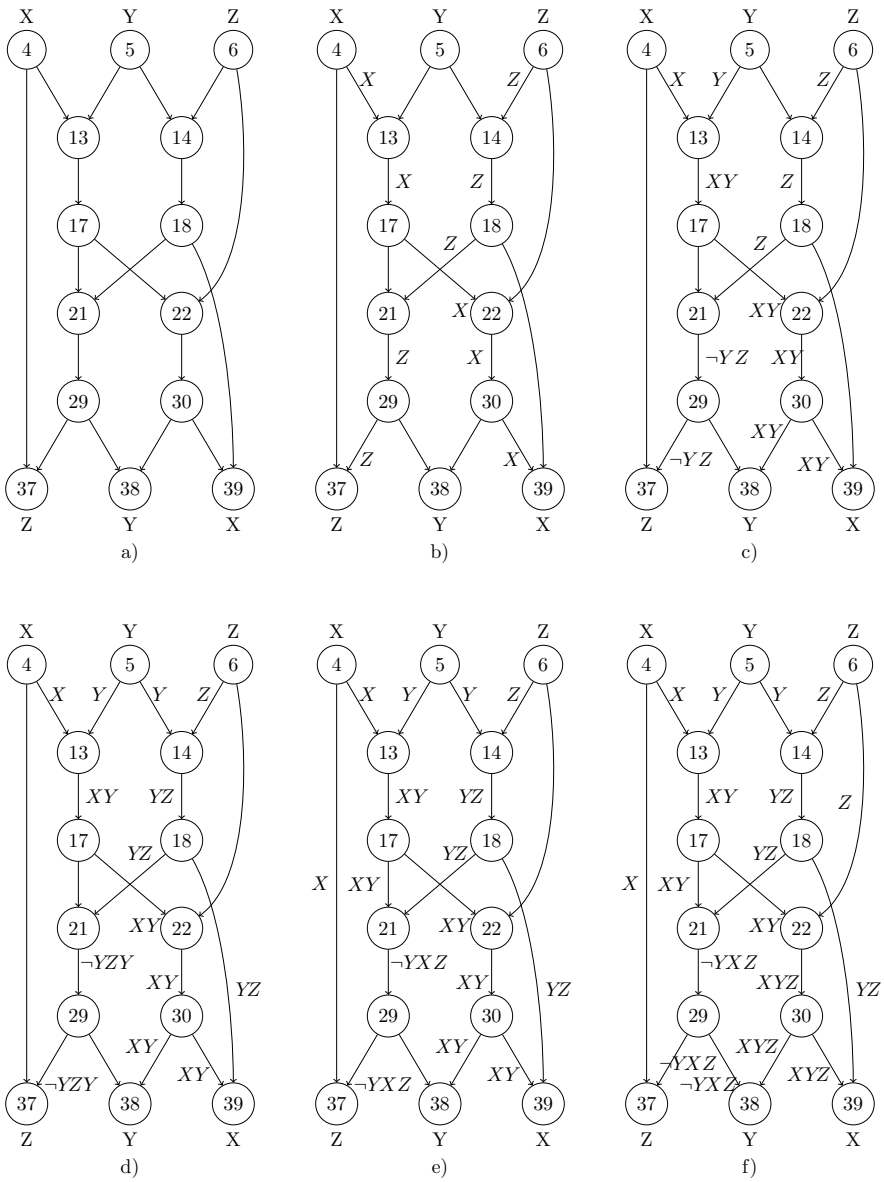
Sætning 5.23:
Betragt en funktion h, der tager funktioner som input, og som er reversibel i alle sine input. Hvis alle disse funktioner er reversible i alle deres variable, så gælder det for alle variabler, der kun optræder i en af funktionerne, at h er reversibel i disse variabel.

Bevis:
Betragt $h(f(X,Y),g(Y,Z),\ldots)$, hvor alle input i h er funktioner af en eller flere variable, som er reversible i alle deres variable, og h er reversibel i alle sine inputs. Antag, at f er den eneste funktion, hvori X indgår. Det ses nu, at hvis vi holder alle andre variabler fast, og kun varierer på X, så må alle andre inputs end f i h være konstant og f må pga. sin reversibilitet ændre værdi for hvert X, når Y holdes fast. Da h er reversibel i f betyder dette, at h ændrer værdi hver gang f ændre værdi, og de andre inputs er konstante. Det betyder, at når vi holder alle andre variabler fast, og ændrer på X, så vil f og dermed h ændre værdi. Dette er definitionen på reversibilitet, og h er dermed reversibel i X. ■

5.4 Eksempel på begrænsning

Vi har nu introduceret en række sætninger, som på to forskellige måder kan hjælpe os til at begrænse mængden af funktioner, som vi skal vælge imellem. På denne måde kan vi forsøge at øge sandsynligheden for at lave et rigtigt valg, så kombinationen af funktioner løser netværkskodningsproblemet.
Sætningerne 5.10, 5.13 og 5.14 samt formodningerne 5.11 og 5.12 er sætninger der omhandler mængden af mulige funktioner som de alle kanter skal vælge imellem. Med disse sætninger kan vi argumentere for at det, at skære mængden af mulige funktioner ned til de ligefordelte eller reversible giver logisk mening, idet de lineære koder er indeholdt i denne mængde. Samtidig kan vi, hvis formodningerne holder, sige, at hvis der er løsninger vil vi stadig have løsninger i denne mængde, og at den procentvise andel af løsninger stiger ved at skære ned til denne mængde.
Sætningerne 5.16 til 5.22 derimod er mere specifikke og opstiller krav til de enkelte kanters valg af funktioner, og begrænser dermed mængden af mulige funktioner, som den enkelte kant kan vælge imellem. Disse sætninger kræver dog langt hen af vejen, at alle mulige funktioner i netværket er reversible. Sætning 5.23 siger noget om at bibeholde denne reversibilitet.
Vi giver nu et eksempel på hvordan disse sætninger kan benyttes til at skære mængden af mulige funktioner ned for hvert enkelt punkt.

Eksempel 5.24:
Vi betragter den venstre del af netværket fra kapitel 5. På figur 5.2 a) ser vi udsnittet af figuren som viser hvad de enkelte afsendere genererer, og hvad modtagerne kræver. Vi ved, at funktionerne i hvert enkelt indkodningspunkt enten kan være nulfunktionen eller en kombination af et eller flere input. Vi antager, at punkter, som kun har ét input, blot videresender dette uændret eller sender nulfunktionen, hvilket betyder, at vi kun skal vælge funktioner i forbindelse med kanterne $e_{13,17}$, $e_{14,18}$, $e_{21,29}$ og $e_{22,30}$.
Idet punkt 37 kræver Z og der kun er en mulig vej fra punkt 6, hvor Z genereres, til punkt 37, ved vi, at funktionerne på disse kanter nødvendigvis må være reversible i Z. Det samme gør sig gældende for punkt 39 som kræver X, som bliver genereret i punkt 4. Det giver, at funktionerne til $e_{13,17}$ og $e_{22,30}$ skal være reversible for X og funktionerne til $e_{14,18}$ og $e_{21,29}$ skal være reversible for Z. Vi indskriver det vi nu ved på figur 5.2 b).
Det ses at punkt 38 kræver Y, men da der findes to kantdisjunkte veje mellem punkt 38 og punkt 5 som generere Y kan vi ikke umiddelbart sige til hvilke



Figur 5.2: *Illustration af trinnene i begrænsningen af mulige funktioner. $\neg Y$ betyder at der på kanten ikke må være Y..*

kanter, funktionerne skal være reversible i Y for at få Y frem til punkt 38. Dog kan vi se, at punkt 37 kun kan få X eller nulfunktionen ind på kanten $e_{4,37}$. Det betyder, at hvis funktionen til $e_{21,29}$ skal være reversibel for Y (således, at der er reversibilitet for Y på $e_{29,38}$), så vil punktet 37 enten modtage en kombination af X eller nulfunktionen og $f(Y,Z)$ eller $f(X,Y,Z)$. Hvis funktionen til $e_{4,37}$ er nulfunktionen kan det åbenlyst ikke lade sig gøre at udlede Z med nogle af de andre inputs, så vi må antage at denne funktion er X. Vi erindrer, at reversibilitet i Y medfører afhængighed af Y, og ser, at i punkt 39 vil vi altså enten have $h(X, f(Y,Z)) = Z$, hvilket ifølge sætning 5.19 ikke kan lade sig gøre, eller $h(X, f(X,Y,Z)) = Z$, hvilket ifølge sætning 5.18 ikke er muligt. Vi kan altså ikke have at funktionen til $e_{21,29}$ er reversibel for Y, hvilket betyder, at der kun er en måde hvorpå Y kan komme fra punkt 5 til punkt 38, hvilket betyder, at funktionerne til $e_{13,17}$ og $e_{22,30}$ skal være reversible for Y. Dette og konsekvenserne af dette ses på figur 5.2 c).
Punkt 39 får nu ikke længere kun besked X ind og vi er derfor nødt til at få noget ind på kant $e_{18,39}$ som kan fjerne den ekstra information som funktionen til kant $e_{30,39}$ har. Vi er altså pr. definition nødt til at have, at kant $e_{18,39}$ har information om besked Y, hvilket betyder, at funktionen til $e_{14,18}$ skal være reversibel for Y. Dette får de konsekvenser vi ser på figur 5.2 d).
Det ses nu, at funktionen til kant $e_{21,29}$ er både reversibel i Y og ikke-reversibel i Y, hvilket er umuligt. Vi er derfor nødt til at have reversibiliteten for Y fjernet i denne funktion. Vi kan fjerne reversibiliteten i Y i funktionen til kanten $e_{21,29}$, hvis den får inputs Y og $g(Y,Z)$. Dette Y kan kun komme fra funktionen til $e_{13,17}$, som derfor også giver reversibilitet i X og vi kræver derfor $h(f(X,Y), g(Y,Z)) = k(X,Z)$, hvilket ifølge sætning 5.17 er muligt. Idet vi fjerner reversibiliteten i Y bliver der altså tilføjet reversibilitet i X, hvilket påvirker punkt 37. For at punkt 37 nu kan dekode Z skal vi pr. definition af reversibilitet, have X ind på kant $e_{4,37}$, hvilket umiddelbart kan lade sig gøre. Alt dette ses på figur 5.2 e).
Lad os nu betragte modtager 39. Vi kan nu ud fra sætning 5.20 sige, at dette punkt ikke på nuværende tidspunkt kan løses. Den eneste mulighed vi har for at ændre situationen er, at lade Z komme ind på kant $e_{30,39}$. Dette betyder ikke nødvendigvis, at modtager 39 nu kan dekode X, men hvis ikke vi gør dette kan den i hvert fald ikke. Vi får nu også Z ned til modtager 38. Denne kan ikke dekode hvis den kun får input $f(X,Y,Z)$ på en kant, hvorfor vi også lader den modtage $f(X,Z)$ på den anden kant. Vi kan ikke sige om dette sikrer en mulig dekodning. Modtager 37 derimod kan vi med sikkerhed sige kan dekode på nuværende tidspunkt. Vi har nu analyseret netværket så meget som vi kan.

Vi har ikke sikret, at der findes en løsning, men vi har opsat nogle krav, som i hvert fald skal opfyldes for at en løsning kan findes.

Vi har nu vist hvordan sætningerne kan bruges til at præciserer hvad de enkelte funktioner skal opfylde i et netværkskodningsproblem. Det er dog ikke i alle tilfælde, at det vil være lige så let at påbegynde en analyse af et netværkskodningsproblem som det var i eksempel 5.24. Det er heller ikke altid muligt, at sætte begrænsninger på alle kanter.
Den metode vi har vist kan bruges til at begrænse mængden af funktioner, som kan bruges til at lave en løsning for et netværkskodningsproblem, men den kan ikke nødvendigvis bruges til hverken at be- eller afkræfte at der findes en løsning til et givent netværkskodningsproblem. Dog kan man i nogle tilfælde vise, at der findes eller ikke findes en reversibel løsning.

Kapitel 6

Eksperimenter

I dette afsnit vil vi undersøge muligheden for, at vi ved at vælge tilfældige funktioner til alle indkodningspunkter kan opnå en løsning på netværkskodningsproblemet fra eksempel 5.2 side 99.
Vi vil undersøge denne sandsynlighed både hvis vi vælger tilfældigt mellem alle mulige funktioner, de ligefordelte funktioner, de reversible funktioner og de lineære funktioner. Vi forstår i dette tilfælde igen en lineær funktion som en funktion på formen $f(X_1, \cdots, X_n) = a_0 + a_1X_1 + \cdots + a_nX_n$, som er afhængig af alle sine variabler.
For hver test er der lavet 1.000.000 eksperimenter hvor funktionerne til alle indkodningspunkter er valgt tilfældigt mellem de tilladte funktioner. Vi har lavet eksperimenter med forskellige alfabetstørrelser for at undersøge hvordan det skalerer. Til hvert eksperiment er der noteret hvor mange tilfældige kombinationer af funktioner, der førte til en løsning af netværkskodningsproblemet og hvor mange, der i hvert fald løste for den første modtager, og hvor mange der løste for de to første modtagere.
Funktioner repræsenteres ved en vektor, som indeholder resultatet af alle kombinationer af inputs. Vi antager for hele kapitlet, at vi arbejder med et alfabet A af størrelse $|A|$ og at antallet af inputs er F.

6.1 Alle funktioner

Disse er de klart nemmeste funktioner at generere, da dette blot kan gøres ved at vælge hver enkelt indgang i funktionsvektoren tilfældigt fra alfabetet.
Vi skal således vælge $|A|^F$ tegn, hvilket kan gøres på $|A|^{(|A|^F)}$ forskellige måder.

Grunden til at der skal vælges $|A|^F$ tegn er, at dette svarer til antallet af mulige input, der kan forekomme. I $\mathbb{F}_5$ er der således $2{,}98 \cdot 10^{17}$ mulige funktioner.

Alle fkt.	Løser for modtager 37	Løser for modtager 37 og 38	Løsninger
$\mathbb{F}_2$	26679	1242	248
$\mathbb{F}_3$	0	0	0
$\mathbb{F}_4$	0	0	0
$\mathbb{F}_5$	0	0	0

Tabel 6.1: *Antal løsninger med alle funktioner.*

I tabel 6.1 ses det, at så snart alfabetstørrelsen stiger, falder sandsynligheden for at finde løsninger meget hurtigt. Hvor der i $\mathbb{F}_2$ bliver fundet ca. 2,6%, der løser netværket for den første modtager, bliver der slet ikke fundet nogen, der løser for første modtager når alfabetstørrelsen er 3 eller højere.

6.2 Ligefordelte funktioner

Disse funktioner genererede vi ved at lave et array af rigtig længde og fylde de første $|A|^{F-1}$ indgange med det første element fra alfabetet og de næste $|A|^{F-1}$ indgange med det andet element i alfabetet og så fremdeles. Når hele vektoren er blevet fyldt op blandes den, og dermed har vi sikret en ligefordelt funktion. Dette svarer til at vi først placerer $|A|^{F-1}$ af det første element fra alfabetet på tilfældige pladser, dette kan gøres på $\binom{|A|^F}{|A|^{F-1}}$ forskellige måder. Derefter skal der placeres $|A|^{F-1}$ af det andet element fra alfabetet, dette kan nu gøres på $\binom{|A|^F-|A|^{F-1}}{|A|^{F-1}}$ måder da nogle af pladserne er optaget af det første element.
Dette giver, at det samlet antal af ligefordelte funktioner kan beregnes på følgende måde,

$$\prod_{i=0\ldots|A|-1} \binom{|A|^F - i \cdot |A|^{F-1}}{|A|^{F-1}}.$$

I $\mathbb{F}_5$ er der således $6{,}23 \cdot 10^{14}$ mulige funktioner.
I tabel 6.2 ses det, at ved $\mathbb{F}_2$ bliver der fundet omkring 20%, der kommer forbi den første modtager og samlet bliver der fundet omkring 1,2%, der løser netværkskodningsproblemet. Når alfabet størrelsen stiger er det igen tydeligt, at antallet af løsninger falder drastisk.

Ligefordelte fkt.	Løser for modt. 37	Løser for modt. 37 og 38	Løsninger
$\mathbb{F}_2$	220250	61072	12153
$\mathbb{F}_3$	27	1	0
$\mathbb{F}_4$	0	0	0
$\mathbb{F}_5$	0	0	0

Tabel 6.2: *Antal løsninger med ligefordelte funktioner.*

6.3 Reversible funktioner

Reversible funktioner er de sværeste at generere, da der er mange ting man skal tage højde for, og på nuværende tidspunkt er det kun muligt for os at generere reversible funktioner med to input, altså for indkodningspunkter med en ind-grad på 2. Se appendix A.
Når vi skal generere reversible funktioner med to input, skal vi tage højde for disse inputs i outputtet. Dette betyder, at hver gang et input holdes fast og det andet varieres, så skal outputtet variere. Dette svarer til at fylde en $|A| \times |A|$-matrix med $|A|$ forskellige symboler, således, at hvert symbol optræder én gang i hver række og én gang i hver søjle. Hver række svarer så til, at den første variabel fastholdes og den anden varieres og hver søjle svarer til, at den anden variabel fastholdes og den første varieres. Det maksimale antal af måder man kan fylde en sådan matrice på er givet ved $\prod_{i=0}^{|A|-1} (|A|-i)^i (|A|-i)!$. Dette skyldes, at der i den første række, kan vælges mellem $|A|$ symboler på første plads, $|A|-1$ symboler på anden plads, osv. - altså $|A|!$. I anden række kan vi på første plads vælge mellem $|A|-1$ symboler, da søjlen allerede indeholder et symbol. På anden plads kan vi igen vælge mellem maksimalt $|A|-1$ symboler, da både rækken og søjlen indeholder et symbol i forvejen, som evt. kan være det samme. På tredje plads er vi nede på at vælge mellem $|A|-2$ symboler, da rækken allerede indeholder to symboler osv. Dette giver $(|A|-1)(|A|-1)!$. I tredje række kan vi på første plads vælge mellem $|A|-2$ symboler, da søjlen indeholder to symboler allerede. På anden plads gør det samme sig gældende, idet det symbol, som er i rækken potentielt kan være magen til et af de symboler, der er i søjlen, så vi har $|A|-2$ muligheder. På tredje plads har vi igen maksimalt $|A|-2$ muligheder, da der er to symboler brugt i både rækken og søjlen, og de kan være ens. På fjerde plads er vi nede på $|A|-3$ muligheder, da der er brugt tre symboler i rækken osv. Denne række har altså $(|A|-2)^2(|A|-2)!$ mulige kombinationer. Antallet af måder at fylde hele matricen på bliver så produktet af antallet af måder hver enkelt række kan

konstrueres på. Alt dette kan illustreres med en matrice, med det maksimale antal af muligheder for hver plads skrevet ind.

$$\begin{array}{c} \\ y_1 \\ y_2 \\ y_3 \\ y_4 \\ \vdots \end{array} \begin{array}{c} \begin{array}{ccccc} x_1 & x_2 & x_3 & x_4 & \cdots \end{array} \\ \begin{bmatrix} |A| & |A|-1 & |A|-2 & |A|-3 & \cdots \\ |A|-1 & |A|-1 & |A|-2 & |A|-3 & \\ |A|-2 & |A|-2 & |A|-2 & |A|-3 & \\ |A|-3 & |A|-3 & |A|-3 & |A|-3 & \\ \vdots & & & & \ddots \end{bmatrix} \end{array}$$

I $\mathbb{F}_5$ er der således maksimalt $9{,}95 \cdot 10^6$ mulige funktioner.

Reversible fkt.	Løser for modt. 37	Løser for modt. 37 og 38	Løsninger
$\mathbb{F}_2$	1000000	1000000	1000000
$\mathbb{F}_3$	500829	250972	0
$\mathbb{F}_4$	10661	437	74
$\mathbb{F}_5$	2	0	0

Tabel 6.3: *Antal løsninger med reversible funktioner.*

I tabel 6.3 ses det, at alle reversible funktioner i $\mathbb{F}_2$ er en løsning. I $\mathbb{F}_3$ løser ca. halvdelen af funktionerne netværkskodningsproblemet for den første modtager og ca. halvdelen af disse løser også for den anden modtager, dog er der ingen af dem, der løser for alle modtagerne. Det er der til gengæld i $\mathbb{F}_4$ hvor 74 af de valgte funktioner løser netværkskodningsproblemet. At vi finder løsninger i $\mathbb{F}_4$ og ikke i $\mathbb{F}_3$ kunne tyde på, at der ikke findes løsninger i $\mathbb{F}_3$.

6.4 Lineære funktioner

For at kunne lave en fornuftig sammenligning har vi også valgt at lave eksperimentet med lineære funktioner, for at kunne se, om der kan være en fordel ved at bruge andet end lineære funktioner selvom de kan løse netværkskodningsproblemet.
Antallet af lineære funktioner beregnes ved $|A|(|A|-1)^F$. I $\mathbb{F}_5$ er der således

80 mulige funktioner. Det ses, at i $\mathbb{F}_2$ løser alle de valgte lineære funktioner netværkskodningsproblemet. I $\mathbb{F}_3$ løser ca. det halve af funktionerne for den første modtager og ca. det halve af dem også for den anden. Ingen løser dog for hele problemet, hvilket ikke er overraskende, da det bevises i artiklen af Dougherty et al. [4], at problemet ikke kan løses med lineære funktioner i legemer med ulige karakteristik. I $\mathbb{F}_4$ løser ca. en tredjedel for første modtager og ca. en tredjedel af disse også for anden modtager og hele systemet. Det ses, at alle funktioner, der løser for anden modtager også løser det hele. I $\mathbb{F}_5$ løser ca. en fjerdedel for første modtager og ca. en fjerdedel af disse også for anden modtager. Igen løser ingen funktioner for det hele, hvilket igen ikke overrasker.

Lineære fkt.	Løser for modt. 37	Løser for modt. 37 og 38	Løsninger
$\mathbb{F}_2$	1000000	1000000	1000000
$\mathbb{F}_3$	499321	249273	0
$\mathbb{F}_4$	333613	111028	111028
$\mathbb{F}_5$	250787	62704	0

Tabel 6.4: *Antal løsninger med lineære funktioner.*

6.5 Sammenligning

Vores eksperimenter tyder på, at sandsynligheden for, at en tilfældigt valgt funktion løser netværkskodningsproblemet stiger hvis man vælger reversible funktioner frem for ligefordelte, hvilke igen er bedre end helt tilfældigt valgte funktioner.
Det skal dog bemærkes at vi kun har lavet vores forsøg på et enkelt netværkskodningsproblem så endnu kan vi ikke sige noget generelt.
Det ses, at i $\mathbb{F}_3$ er antallet af løsninger stort set ens for lineære og reversible funktioner. Dette stemmer godt overens med, at vi har vist, at alle reversible funktioner i $\mathbb{F}_3$ er lineære.

Kapitel 7

Konklusion

Vi har i dette speciale arbejdet med forskellige emner indenfor netværkskodning. Vi har betragtet den helt generelle teori, og undersøgt hvad det betyder for et netværkskodningsproblem at være løseligt. Vi studerede multicast netværkskodningsproblemer og fandt ud af, at et multicast netværkskodningsproblem med h beskeder kan løses hvis og kun hvis det der eksisterer et flow af størrelse h fra mængden af afsendere til hver enkelt modtager og hvis det kunne løses eksisterer der en lineær løsning.
Vi fortsatte herfra til det lineære multicast netværkskodningsproblem og konstaterede, at for dette problem kunne vi opstille en række matricer ud fra hvilke vi kunne lave det såkaldte transferpolynomium, som indikerer om problemet kan løses. Det er desuden muligt for et løseligt lineært multicast netværkskodningsproblem at finde en løsning ved blot at vælge tilfældige funktioner i alle indkodningspunkter, hvis legemet er stort nok
Vi betragtede herefter underrumskoder for lineære multicast netværkskodningsproblemer. Nærmere bestemt kiggede vi på Kötter-Kschischang koder, som er fejl- og sletningskorrigerende koder. Disse koder viste sig at være rigtig gode til at håndtere fejl for så vidt, at en fejl ikke kan propagere i netværket og derfor ikke spreder sig. Desværre viste det sig at en fejl kunne lede til en sletning og altså dermed gav to problemer. Dette kan måske løses ved at forsøge at benytte en lineær kode sammen med Kötter-Kschischang koder og dermed benytte de lineære koder til fejlretning og Kötter-Kschischang koderne til sletninger. Dette er dog udenfor rapportens fokus, men kan foreslås til videre arbejde.
Vi betragtede herefter lineær netværkskodning generelt og forsøgte at udvide teorien fra multicast til også at gælde i dette tilfælde. Vi opstillede de samme

matricer, men fik i dette tilfælde ud over et transferpolynomium også en mængde ligninger for støj. For at netværkskodningsproblemet kunne være løseligt skulle transferpolynomiet være ikke-nul mens støjligningerne var nul. Dette ligningssystem knyttede vi til Gröbnerbasisteori og fandt ud af at forbinde netværkskodningsproblemets løselighed til hvorvidt den tilhørende Gröbnerbasis indeholdt tallet et. Der er stadig mange åbne spørgsmål indenfor dette emne, hvilket vi foreslår til videre arbejde. Der er en del huller i teorien for så vidt det drejer sig om sandsynligheden for at finde en korrekt løsning ved tilfældigt valg og hvordan denne sandsynlighed relaterer sig til legemets størrelse. Samtidig kan det være interessant at betragte den topologiske mening af flere af ligningerne, der fremkommer i forbindelse med at finde en Gröbnerbasis. Samt, for et netværkskodningsproblem, der kan løses lineært, at se hvad den reducerede Gröbnerbasis består af for nogle ligninger og deres eventuelle forhold til netværkets topologi. Desuden kan det undersøges om den reducerede Gröbnerbasis kan sige noget om antallet af løsninger.
Til sidst betragtede vi det generelle netværkskodningsproblem. Idet nogle netværkskodningsproblemer ikke kan løses med lineær netværkskodning forsøgte vi at finde metoder til at finde løsninger til disse netværk. Vi forsøgte, som ved lineære multicast netværkskodningsproblemer at gætte løsninger tilfældigt, men dette var tilsyneladende ikke effektivt, når vi valgte mellem alle mulige funktioner. Vi forsøgte herefter at begrænse mængden af funktioner, som vi valgte imellem uden at tage hensyn til netværkets topologi, hvilket gav forbedrede resultater. Vi forsøgte herefter yderligere at begrænse mængden af funktioner ved at analysere netværket og opstille krav til de funktioner, som skulle bruges i de enkelte punkter. Som videre arbejde foreslår vi en større gennemtestning af hvorvidt det hjælper at begrænse funktionsmængden både på den ene og den anden måde, samt beregning og bevis af hvordan sandsynligheden er for at finde en løsning tilfældigt. Det står os ikke helt klart hvorvidt begrænsningen af funktioner også fjerner en del løsninger, hvilket selvfølgelig påvirker sandsynligheden. Dette bør undersøges.

Bilag A

Latinske kvadrater

Dette kapitels grundidé er baseret på en artikel af Jacobson and Matthews [9]. For at gøre det muligt for os at generere de reversible funktioner vi bruger i kapitel 6 til at lave vores forsøg med, var vi nødt til at finde en måde at lave disse matricer, hvor hvert element i alfabetet forekommer præcist en gang i hver række og søjle. Dette svarer til at skulle generere latinske kvadrater.
Lad os starte med, at definere et latinsk kvadrat.

Definition A.1 (Latinsk kvadrat):
Et latinsk kvadrat af orden n er en $n \times n$-matrix med n forskellige elementer i indgangene, hvor alle elementer optræder én gang i hver række og én gang i hver søjle.

Lad os betragte et eksempel.

Eksempel A.2:
Et 4×4-latinsk kvadrat.

$$\begin{bmatrix} a & b & c & d \\ b & d & a & c \\ d & c & b & a \\ c & a & d & b \end{bmatrix}$$

I disse kvadrater ønsker vi at være i stand til at foretage transformationer, for at bevæge os fra et latinsk kvadrat til et andet.

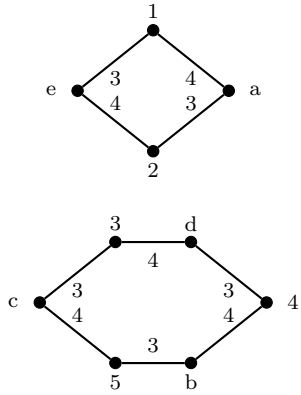
Definition A.3 (Skridt):
Vi definerer et skridt, som værende en form for transformation i et latinsk kvadrat. For alle typer af skridt gælder, at vi med et eller flere skridt af samme type igen kan komme frem til et latinsk kvadrat.

Vi noterer os altså, at vi i nogle tilfælde bruger flere skridt før vi når et latinsk kvadrat. De matricer, som opstår ved at foretage et skridt, som ikke leder til et latinsk kvadrat kalder vi uegentlige latinske kvadrater. For at kunne kende forskel kalder vi herefter latinske kvadrater for egentlige latinske kvadrater, hvis der kan være tvivl om hvad der menes.

Lad os nu prøve at introducere nogle forskellige former for skridt, som kan foretages i et latinsk kvadrat. To oplagte former for skridt er at ombytte to rækker, søjler eller symboler, hvilket i alle tilfælde vil resultere i et nyt latinsk kvadrat. En anden type af skridt er at ombytte symbolerne i et udsnit af et latinsk kvadrat - hvilket vi kalder en kredsombytning. Vi har i dette tilfælde brug for at definere hvilke typer af udsnit, der kan bruges til disse ombytninger. Vi starter med en række-par graf, som fuldstændigt beskriver forholdet mellem to rækker i et latinsk kvadrat. Lad os først betragte et eksempel.

Eksempel A.4:
Lad os betragte følgende latinske kvadrat og tegne en række-par graf for rækkerne 3 og 4.

$$\begin{bmatrix} b & d & e & c & a \\ d & c & b & a & e \\ e & a & c & d & b \\ a & e & d & b & c \\ c & b & a & e & d \end{bmatrix}$$



Det ses i dette tilfælde, at grafen kan deles op i to disjunkte delgrafer, som hver især er en kreds. Vi siger derfor, at det latinske kvadrat har to kredse.

Vi er nu klar til at lave en formel definition.

Definition A.5 (Række-par graf):
Betragt et latinsk kvadrat af orden n. En række-par graf for to rækker k og l er en to-delelig graf, som består af $2n$ punkter og $2n$ kanter. Den ene uafhængige punktmængde består af n punkter, som repræsenterer de n søjler i det latinske kvadrat, og den anden uafhængige punktmængde består af de n resterende punkter, som repræsenterer de n symboler. De n første kanter repræsenterer rækken k, og de n resterende kanter repræsenterer rækken l. Hvis der i det latinske kvadrat er symbolet a på plads $(k, 4)$, så er punkt a og 4 forbundne med kant k i grafen.

Vi kan på samme måde definere en søjle-par graf og en symbol-par graf.
Vi kan nu definere en kredsombytning.

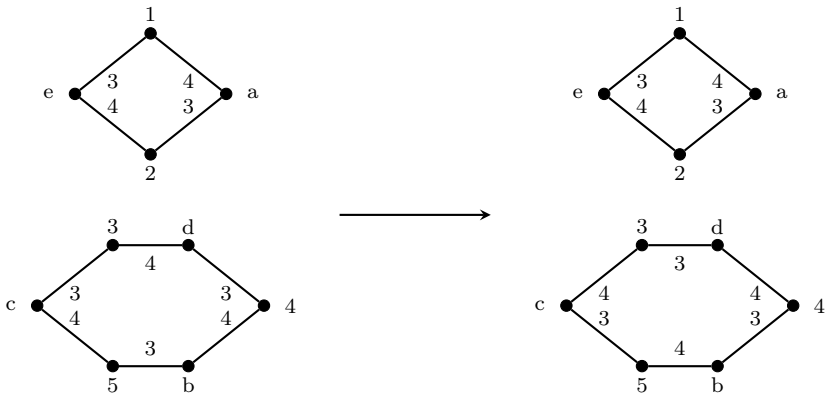
Definition A.6 (Kredsombytning):
En kredsombytning er et skridt i et latinsk kvadrat, hvor vi i en kreds i en række-par (eller søjle-par eller symbol-par) graf ombytter alle række

(respektivt søjle eller symbol) labels på kanterne og ombytter i det latinske kvadrat tilsvarende. Dette fører altid til et nyt egentligt latinsk kvadrat.

Vi kan nu betragte et eksempel på en kredsombytning.

Eksempel A.7:

$$\begin{bmatrix} b & d & e & c & a \\ d & c & b & a & e \\ e & a & c & d & b \\ a & e & d & b & c \\ c & b & a & e & d \end{bmatrix} \rightarrow \begin{bmatrix} b & d & e & c & a \\ d & c & b & a & e \\ e & a & d & b & c \\ a & e & c & d & b \\ c & b & a & e & d \end{bmatrix}$$



De former for skridt, som vi hidtil har indført er dog ikke tilstrækkelige til at kunne nå alle mulige latinske kvadrater, da den generelle struktur bibeholdes. F.eks. ses det ovenstående kredsombytning, at der er to kredse i rækkerne 3 og 4. Nemlig kredsen $\{a, e\}$ og kredsen $\{b, c, d\}$. Lige meget hvordan vi ombytter rækker, søjler eller symboler i kredsene vil der stadig eksistere disse to kredse i disse rækker. Denne generelle struktur fastholdes altså.
Lad os nu definere en klasse af skridt, ± 1-skridt, som kan bryde disse strukturer. Dette er en form for ombytninger i et latinsk kvadrat, hvor elementer

ombyttes. Dette betyder, at vi i første omgang ofte kommer til uegentlige latinske kvadrater, men disse uegentlige latinske kvadrater er skridt på vejen mellem de egentlige latinske kvadrater.
Først laver vi en alternativ repræsentation af latinske kvadrater. Det gælder, at et latinsk kvadrat af orden n er ækvivalent med en $n \times n \times n$- (0-1)-matrix, hvis dimensioner svarer til rækkerne, søjlerne og elementerne i den tilsvarende latinske kvadrat. Matricen er fyldt således, at der er et 1-tal på indgang (r, s, e) hvis og kun hvis elementet e optræder i række r, søjle s i den tilsvarende latinske kvadrat ellers er indgangen 0 i matricen. Vi kalder en sådan matrix for *forekomstmatricen*.

Eksempel A.8:
Lad os betragte et eksempel på en forekomstmatrix. Vi har følgende latinske kvadrat af orden 5 med elementerne $\{a, b, c, d, e\}$.

$$\begin{bmatrix} a & d & b & c & e \\ e & c & d & a & b \\ b & a & e & d & c \\ c & e & a & b & d \\ d & b & c & e & a \end{bmatrix}.$$

Denne svarer til følgende forekomstmatrix, hvor de enkelte lag er lagt ud ved siden af hinanden.

element a					element b					element c					element d					element e				
1	0	0	0	0	0	0	**1**	0	0	0	0	0	0	**1**	0	0	**1**	0	0	0	0	0	0	0	**1**
0	0	0	**1**	0	0	0	0	0	**1**	0	**1**	0	0	0	0	0	**1**	0	0	**1**	0	0	0	0	0
0	**1**	0	0	0	**1**	0	0	0	0	0	0	0	0	**1**	0	0	0	**1**	0	0	0	**1**	0	0	0
0	0	**1**	0	0	0	0	0	**1**	0	**1**	0	0	0	0	0	0	0	0	**1**	0	**1**	0	0	0	0
0	0	0	0	**1**	0	**1**	0	0	0	0	0	**1**	0	0	**1**	0	0	0	0	0	0	0	**1**	0	0

Det ses altså, at i hvert lag optræder der ét ettal i hver række og ét i hver søjle. Det ses desuden, at der kun optræder ét ettal på den samme plads i lagene.

Lad os nu forklare ± 1-skridt. I forekomstmatricen er alle linjesummer 1. Vi forholder os nu til disse linjesummer og udvælger en $2 \times 2 \times 2$- delmatrix, hvori vi i hver indgang indsætter +1 eller -1 således, at linjesummerne er uændrede. Dette kaldes et ± 1-skridt.

Definition A.9 (±1-skridt):
Et ±1-skridt er en operation på en $2 \times 2 \times 2$-matrix således, at der i hver indgang lægges en til eller trækkes en fra fordelt således, at der i 4 indgange lægges til og i 4 indgange trækkes fra. Dette gøres på en måde således, at der i alle rækker, søjler og lag hhv lægges en til og trækkes en fra således, at linjesummen bevares.

Da hver enkelt linje i matricen skal have indsat et +1 og et -1, må vi altså nedskrive 4 indgange. Disse kan potentielt være nulindgange, hvilket betyder, at de nu er negative. Vi kan begrænse antallet af negative indgange på følgende måde. Vælg en indgang i forekomstmatricen, som er nul. Denne ligger i tre linjer, som alle indeholder et ettal. Lad nu de indgange, som indeholder et ettal definere delmatricen. Denne delmatrix vil alt efter valget af nulindgang enten bestå af 4 ettaller og 4 nuller eller af 3 ettaller og 5 nuller. Dette er logisk, hvis man husker, at der kun er et ettal i hver linje. I følgende forekomstmatrix, hvor vi først vælger en nulindgang (med cirkel omkring) og derefter en anden ses de to typer af delmatricer.

element a	element b	element c	element d	element e
$\begin{bmatrix} \mathbf{1} & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & \mathbf{1} & 0 \\ 0 & \boxed{\mathbf{1}} & \boxed{0} & 0 & 0 \\ 0 & \boxed{\textcircled{0}} & \boxed{\mathbf{1}} & 0 & 0 \\ 0 & 0 & 0 & 0 & \mathbf{1} \end{bmatrix}$	$\begin{bmatrix} 0 & 0 & \mathbf{1} & 0 & 0 \\ 0 & 0 & 0 & 0 & \mathbf{1} \\ \mathbf{1} & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & \mathbf{1} & 0 \\ 0 & \mathbf{1} & 0 & 0 & 0 \end{bmatrix}$	$\begin{bmatrix} 0 & 0 & 0 & \mathbf{1} & 0 \\ 0 & \mathbf{1} & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & \mathbf{1} \\ \mathbf{1} & 0 & 0 & 0 & 0 \\ 0 & 0 & \mathbf{1} & 0 & 0 \end{bmatrix}$	$\begin{bmatrix} 0 & \mathbf{1} & 0 & 0 & 0 \\ 0 & 0 & \mathbf{1} & 0 & 0 \\ 0 & 0 & 0 & \mathbf{1} & 0 \\ 0 & 0 & 0 & 0 & \mathbf{1} \\ \mathbf{1} & 0 & 0 & 0 & 0 \end{bmatrix}$	$\begin{bmatrix} 0 & 0 & 0 & 0 & \mathbf{1} \\ \mathbf{1} & 0 & 0 & 0 & 0 \\ 0 & \boxed{0} & \boxed{\mathbf{1}} & 0 & 0 \\ 0 & \boxed{\mathbf{1}} & \boxed{0} & 0 & 0 \\ 0 & 0 & 0 & \mathbf{1} & 0 \end{bmatrix}$

Tabel A.1: *Eksempel på delmatrice med 4 ettaller og 4 nuller*

element a	element b	element c	element d	element e
$\begin{bmatrix} \mathbf{1} & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & \mathbf{1} & 0 \\ 0 & \mathbf{1} & 0 & 0 & 0 \\ 0 & 0 & \mathbf{1} & 0 & 0 \\ 0 & 0 & 0 & 0 & \mathbf{1} \end{bmatrix}$	$\begin{bmatrix} 0 & 0 & \mathbf{1} & 0 & 0 \\ 0 & 0 & 0 & 0 & \mathbf{1} \\ \mathbf{1} & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & \mathbf{1} & 0 \\ 0 & \mathbf{1} & 0 & 0 & 0 \end{bmatrix}$	$\begin{bmatrix} 0 & 0 & 0 & \mathbf{1} & 0 \\ 0 & \mathbf{1} & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & \mathbf{1} \\ \mathbf{1} & 0 & 0 & 0 & 0 \\ 0 & 0 & \mathbf{1} & 0 & 0 \end{bmatrix}$	$\begin{bmatrix} 0 & \mathbf{1} & 0 & 0 & 0 \\ 0 & 0 & \boxed{\mathbf{1}} & \boxed{0} & 0 \\ 0 & 0 & \boxed{\textcircled{0}} & \boxed{\mathbf{1}} & 0 \\ 0 & 0 & 0 & 0 & \mathbf{1} \\ \mathbf{1} & 0 & 0 & 0 & 0 \end{bmatrix}$	$\begin{bmatrix} 0 & 0 & 0 & 0 & \mathbf{1} \\ \mathbf{1} & 0 & \boxed{0} & \boxed{0} & 0 \\ 0 & 0 & \boxed{\mathbf{1}} & \boxed{0} & 0 \\ 0 & \mathbf{1} & 0 & 0 & 0 \\ 0 & 0 & 0 & \mathbf{1} & 0 \end{bmatrix}$

Tabel A.2: *Eksempel på delmatrice med 3 ettaller og 5 nuller*

Det ses altså, at den indgang, som er ukendt er den indgang, som er modsat

den valgte nulindgang, altså den indgang, som ligger i den anden række, den anden søjle og det andet lag.
Vi laver nu vores ± 1-skridt ved at tilføje $+1$ til de fire kendte nulindgange og -1 til de andre fire indgange. Herved skaber vi potentielt en enkelt -1-indgang. Hvis vores forekomstmatrix nu indeholder en -1-indgang kalder vi denne for en uegentlig forekomstmatrix. Vi er naturligvis interesseret i egentlige forekomstmatricer, men disse uegentlige forekomstmatricer er skridt på vejen til nye egentlige forekomstmatricer.
Fra enhver egentlig forekomstmatrix er der $n^2(n-1)$ mulige ± 1-skridt, svarende til antallet af nulindgange i matricen. Alle ± 1-skridt på forskellige nulindgange producerer forskellige matricer, medmindre nulindgangene indgår i den samme delmatrix med fire ettaller - en sådan delmatrix kan vælges på fire forskellige måder, hvilke alle leder til samme nye egentlig forekomstmatrix.
Fra en egentlig forekomstmatrix kan vi altså lave et ± 1-skridt og komme til en egentlig forekomstmatrix, hvis der er fire ettaller i den valgte delmatrix, eller vi kan komme til en uegentlig forekomstmatrix. Lad os nu se hvordan vi kommer videre fra en uegentlig forekomstmatrix. Vi fokuserer på -1-indgangen, som ligger i tre linjer i matricen i hvilke der er præcis to ettaller (linjesummen er stadig 1). Vi vælger tilfældigt en af 1-indgangene i hver linje, hvilket igen udgør en $2 \times 2 \times 2$- delmatrix. Vi laver et ± 1-skridt, der ændrer -1-indgangen til en 0-indgang. På samme måde som før er det kun den indgang, som er modsat -1-indgangen, som er ukendt. Hvis denne er en 1-indgang laver dette ± 1-skridt en egentlig forekomstmatrix ellers er det en 0-indgang og bliver nu den nye -1-indgang i en ny uegentlig forekomstmatrix.
Vi vælger, at begrænse os til denne slags ± 1-skridt, selvom der findes andre, f.eks. skridt, som skubber -1-indgangen til siden - dette kan opnås med to af vores ± 1-skridt.
Lad os forklare hvordan vi laver en række af ± 1-skridt fra et egentligt latinsk kvadrat til en anden med et eksempel.

Eksempel A.10:
Lad os betragte følgende latinske kvadrat af orden 5.

$$\begin{bmatrix} a & b & c & d & e \\ b & c & d & e & a \\ c & d & e & a & b \\ d & e & a & b & c \\ e & a & b & c & d \end{bmatrix}$$

Vi repræsenterer nu dette latinske kvadrat ved dens forekomstmatrix og manipulerer denne med ± 1-skridt. Vi markerer ændringer med kvadrater. Den første tilfældigt valgte 0-indgang markeres med en cirkel.

	element a	element b	element c	element d	element e
1)	$\begin{bmatrix} \mathbf{1} & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & \mathbf{1} \\ 0 & 0 & 0 & \mathbf{1} & 0 \\ 0 & 0 & \mathbf{1} & 0 & 0 \\ 0 & \mathbf{1} & 0 & 0 & 0 \end{bmatrix}$	$\begin{bmatrix} 0 & \mathbf{1} & 0 & 0 & 0 \\ \mathbf{1} & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & \mathbf{1} \\ 0 & 0 & 0 & \mathbf{1} & 0 \\ 0 & 0 & \mathbf{1} & 0 & 0 \end{bmatrix}$	$\begin{bmatrix} 0 & 0 & \mathbf{1} & 0 & 0 \\ 0 & \mathbf{1} & 0 & 0 & 0 \\ \mathbf{1} & 0 & \textcircled{0} & 0 & 0 \\ 0 & 0 & 0 & 0 & \mathbf{1} \\ 0 & 0 & 0 & \mathbf{1} & 0 \end{bmatrix}$	$\begin{bmatrix} 0 & 0 & 0 & \mathbf{1} & 0 \\ 0 & 0 & \mathbf{1} & 0 & 0 \\ 0 & \mathbf{1} & 0 & 0 & 0 \\ \mathbf{1} & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & \mathbf{1} \end{bmatrix}$	$\begin{bmatrix} 0 & 0 & 0 & 0 & \mathbf{1} \\ 0 & 0 & 0 & \mathbf{1} & 0 \\ 0 & 0 & \mathbf{1} & 0 & 0 \\ 0 & \mathbf{1} & 0 & 0 & 0 \\ \mathbf{1} & 0 & 0 & 0 & 0 \end{bmatrix}$
2)	$\begin{bmatrix} \mathbf{1} & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & \mathbf{1} \\ 0 & 0 & 0 & \mathbf{1} & 0 \\ 0 & 0 & \mathbf{1} & 0 & 0 \\ 0 & \mathbf{1} & 0 & 0 & 0 \end{bmatrix}$	$\begin{bmatrix} 0 & \mathbf{1} & 0 & 0 & 0 \\ \mathbf{1} & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & \mathbf{1} \\ 0 & 0 & 0 & \mathbf{1} & 0 \\ 0 & 0 & \mathbf{1} & 0 & 0 \end{bmatrix}$	$\begin{bmatrix} \boxed{\mathbf{1}} & 0 & \boxed{0} & 0 & 0 \\ 0 & \mathbf{1} & 0 & 0 & 0 \\ \boxed{0} & 0 & \boxed{\mathbf{1}} & 0 & 0 \\ 0 & 0 & 0 & 0 & \mathbf{1} \\ 0 & 0 & 0 & \mathbf{1} & 0 \end{bmatrix}$	$\begin{bmatrix} 0 & 0 & 0 & \mathbf{1} & 0 \\ 0 & 0 & \mathbf{1} & 0 & 0 \\ 0 & \mathbf{1} & 0 & 0 & 0 \\ \mathbf{1} & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & \mathbf{1} \end{bmatrix}$	$\begin{bmatrix} \boxed{\mathbf{-1}} & 0 & \boxed{\mathbf{1}} & 0 & \mathbf{1} \\ 0 & 0 & 0 & \mathbf{1} & 0 \\ \boxed{\mathbf{1}} & 0 & \boxed{0} & 0 & 0 \\ 0 & \mathbf{1} & 0 & 0 & 0 \\ \mathbf{1} & 0 & 0 & 0 & 0 \end{bmatrix}$
3)	$\begin{bmatrix} \boxed{0} & 0 & \boxed{\mathbf{1}} & 0 & 0 \\ 0 & 0 & 0 & 0 & \mathbf{1} \\ 0 & 0 & 0 & \mathbf{1} & 0 \\ 0 & 0 & \mathbf{1} & 0 & 0 \\ \boxed{\mathbf{1}} & \mathbf{1} & \boxed{\mathbf{-1}} & 0 & 0 \end{bmatrix}$	$\begin{bmatrix} 0 & \mathbf{1} & 0 & 0 & 0 \\ \mathbf{1} & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & \mathbf{1} \\ 0 & 0 & 0 & \mathbf{1} & 0 \\ 0 & 0 & \mathbf{1} & 0 & 0 \end{bmatrix}$	$\begin{bmatrix} \mathbf{1} & 0 & 0 & 0 & 0 \\ 0 & \mathbf{1} & 0 & 0 & 0 \\ 0 & 0 & \mathbf{1} & 0 & 0 \\ 0 & 0 & 0 & 0 & \mathbf{1} \\ 0 & 0 & 0 & \mathbf{1} & 0 \end{bmatrix}$	$\begin{bmatrix} 0 & 0 & 0 & \mathbf{1} & 0 \\ 0 & 0 & \mathbf{1} & 0 & 0 \\ 0 & \mathbf{1} & 0 & 0 & 0 \\ \mathbf{1} & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & \mathbf{1} \end{bmatrix}$	$\begin{bmatrix} \boxed{0} & 0 & \boxed{0} & 0 & \mathbf{1} \\ 0 & 0 & 0 & \mathbf{1} & 0 \\ \mathbf{1} & 0 & 0 & 0 & 0 \\ 0 & \mathbf{1} & 0 & 0 & 0 \\ \boxed{0} & 0 & \boxed{\mathbf{1}} & 0 & 0 \end{bmatrix}$
4)	$\begin{bmatrix} 0 & 0 & \mathbf{1} & 0 & 0 \\ 0 & 0 & 0 & 0 & \mathbf{1} \\ 0 & 0 & 0 & \mathbf{1} & 0 \\ 0 & \boxed{\mathbf{1}} & \boxed{0} & 0 & 0 \\ \mathbf{1} & \boxed{0} & \boxed{0} & 0 & 0 \end{bmatrix}$	$\begin{bmatrix} 0 & \mathbf{1} & 0 & 0 & 0 \\ \mathbf{1} & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & \mathbf{1} \\ 0 & \boxed{\mathbf{-1}} & \boxed{\mathbf{1}} & \mathbf{1} & 0 \\ 0 & \boxed{\mathbf{1}} & \boxed{0} & 0 & 0 \end{bmatrix}$	$\begin{bmatrix} \mathbf{1} & 0 & 0 & 0 & 0 \\ 0 & \mathbf{1} & 0 & 0 & 0 \\ 0 & 0 & \mathbf{1} & 0 & 0 \\ 0 & 0 & 0 & 0 & \mathbf{1} \\ 0 & 0 & 0 & \mathbf{1} & 0 \end{bmatrix}$	$\begin{bmatrix} 0 & 0 & 0 & \mathbf{1} & 0 \\ 0 & 0 & \mathbf{1} & 0 & 0 \\ 0 & \mathbf{1} & 0 & 0 & 0 \\ \mathbf{1} & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & \mathbf{1} \end{bmatrix}$	$\begin{bmatrix} 0 & 0 & 0 & 0 & \mathbf{1} \\ 0 & 0 & 0 & \mathbf{1} & 0 \\ \mathbf{1} & 0 & 0 & 0 & 0 \\ 0 & \mathbf{1} & 0 & 0 & 0 \\ 0 & 0 & \mathbf{1} & 0 & 0 \end{bmatrix}$
5)	$\begin{bmatrix} 0 & 0 & \mathbf{1} & 0 & 0 \\ 0 & 0 & 0 & 0 & \mathbf{1} \\ 0 & 0 & 0 & \mathbf{1} & 0 \\ 0 & \mathbf{1} & 0 & 0 & 0 \\ \mathbf{1} & 0 & 0 & 0 & 0 \end{bmatrix}$	$\begin{bmatrix} 0 & \mathbf{1} & 0 & 0 & 0 \\ \mathbf{1} & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & \mathbf{1} \\ 0 & \boxed{0} & \boxed{0} & \mathbf{1} & 0 \\ 0 & \boxed{0} & \boxed{\mathbf{1}} & 0 & 0 \end{bmatrix}$	$\begin{bmatrix} \mathbf{1} & 0 & 0 & 0 & 0 \\ 0 & \mathbf{1} & 0 & 0 & 0 \\ 0 & 0 & \mathbf{1} & 0 & 0 \\ 0 & 0 & 0 & 0 & \mathbf{1} \\ 0 & 0 & 0 & \mathbf{1} & 0 \end{bmatrix}$	$\begin{bmatrix} 0 & 0 & 0 & \mathbf{1} & 0 \\ 0 & 0 & \mathbf{1} & 0 & 0 \\ 0 & \mathbf{1} & 0 & 0 & 0 \\ \mathbf{1} & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & \mathbf{1} \end{bmatrix}$	$\begin{bmatrix} 0 & 0 & 0 & 0 & \mathbf{1} \\ 0 & 0 & 0 & \mathbf{1} & 0 \\ \mathbf{1} & 0 & 0 & 0 & 0 \\ 0 & \boxed{0} & \boxed{\mathbf{1}} & 0 & 0 \\ 0 & \boxed{\mathbf{1}} & \boxed{0} & 0 & 0 \end{bmatrix}$

I 1) udvælges den nulindgang, som vi vil arbejde med. Vi udvælger nu en $2 \times 2 \times 2$-matrix ved at lade det udvalgte 0 samt de ettaller, der befinder sig i samme søjle, række eller placering i laget, udgøre hjørnerne. I denne kube foretages vores ± 1-skridt. Resultatet af dette ses i 2). Vi udvælger nu en ny $2 \times 2 \times 2$-matrix ud fra -1-indgangen samt tilfældigt valgte ettaller i samme søjle, række og placering i laget. I denne laves et ± 1-skridt, hvilket ses i 3). Denne manøvre gentages to gange, hvilket giver 4) og 5). Vi er nu kommet

frem til en egentlig forekomstmatrix, som svarer til følgende latinske kvadrat.

$$\begin{bmatrix} c & b & a & d & e \\ b & c & d & e & a \\ e & d & c & a & b \\ d & a & e & b & c \\ a & e & b & c & d \end{bmatrix}$$

Det er ikke helt ligetil at generere et latinsk kvadrat af høj orden, men det viser sig, at hvis man har genereret et latinsk kvadrat, så kan man komme hen til alle andre latinske kvadrater i et endeligt antal skridt.
Følgende sætning omkring disse forbindelser er bevist af Jacobson og Matthews [9].

Sætning A.11:
Givet to (egentlige eller uegentlige) latinske kvadrater af orden n eksisterer der en sekvens af ± 1*-skridt der transformerer den ene matrix om til den anden. En øvre grænse for antallet af skridt i den korteste sekvens er* $2(n-1)^3$*, når* $n \geq 2$*.*

Dette betyder, at hvis vi starter med et latinsk kvadrat og derefter tager $2(n-1)^3$ ± 1-skridt fra denne, så kan vi ende på et hvilken som helst andet latinsk kvadrat.
Ifølge Jacobson og Matthews [9] vil fordelingen af latinske kvadrater, som genereres af en række skridt ud fra samme latinske kvadrat være noget nær uniform.

A.1 Latinske kuber og hyperkuber

Teorien for at generere latinske kvadrater uniformt fordelt er altså kendt, og vi kan derfor anvende denne teori til at generere tilfældige funktioner, som er reversible i to input. Vi har dog også brug for, at generere funktioner, der er reversible i et større antal inputs, hvorfor vi ønsker at udvide denne teori først til latinske kuber og derefter til latinske hyperkuber af vilkårlig orden. Hvis vi betragter en latinsk kube af orden n, kan vi se, at denne stadig kan repræsenteres ved en forekomstmatrix, som dog nu skal være en $n \times n \times n \times n$-matrix. Lad os betragte et eksempel.

Eksempel A.12:
Betragte følgende latinske kube af orden 5, hvor lagene i kuben er lagt ved siden af hinanden.

$$
\begin{array}{ccccc}
\text{lag 1} & \text{lag 2} & \text{lag 3} & \text{lag 4} & \text{lag 5} \\
\left[\begin{array}{ccccc} a & b & c & d & e \\ b & c & d & e & a \\ c & d & e & a & b \\ d & e & a & b & c \\ e & a & b & c & d \end{array}\right] &
\left[\begin{array}{ccccc} b & c & d & e & a \\ c & d & e & a & b \\ d & e & a & b & c \\ e & a & b & c & d \\ a & b & c & d & e \end{array}\right] &
\left[\begin{array}{ccccc} c & d & e & a & b \\ d & e & a & b & c \\ e & a & b & c & d \\ a & b & c & d & e \\ b & c & d & e & a \end{array}\right] &
\left[\begin{array}{ccccc} d & e & a & b & c \\ e & a & b & c & d \\ a & b & c & d & e \\ b & c & d & e & a \\ c & d & e & a & b \end{array}\right] &
\left[\begin{array}{ccccc} e & a & b & c & d \\ a & b & c & d & e \\ b & c & d & e & a \\ c & d & e & a & b \\ d & e & a & b & c \end{array}\right]
\end{array}
$$

Denne latinske kube kan repræsenteres ved følgende forekomstmatrix, hvor lagene i de enkelte kuber er lagt ved siden af hinanden og kuberne, der repræsenterer de enkelte lag i vores latinske kube er under hinanden.

$$
\begin{array}{ccccc}
 & \text{element a} & \text{element b} & \text{element c} & \text{element d} & \text{element e} \\
\text{lag 1} &
\left[\begin{array}{ccccc} \mathbf{1} & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & \mathbf{1} \\ 0 & 0 & 0 & \mathbf{1} & 0 \\ 0 & 0 & \mathbf{1} & 0 & 0 \\ 0 & \mathbf{1} & 0 & 0 & 0 \end{array}\right] &
\left[\begin{array}{ccccc} 0 & \mathbf{1} & 0 & 0 & 0 \\ \mathbf{1} & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & \mathbf{1} \\ 0 & 0 & 0 & \mathbf{1} & 0 \\ 0 & 0 & \mathbf{1} & 0 & 0 \end{array}\right] &
\left[\begin{array}{ccccc} 0 & 0 & \mathbf{1} & 0 & 0 \\ 0 & \mathbf{1} & 0 & 0 & 0 \\ \mathbf{1} & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & \mathbf{1} \\ 0 & 0 & 0 & \mathbf{1} & 0 \end{array}\right] &
\left[\begin{array}{ccccc} 0 & 0 & 0 & \mathbf{1} & 0 \\ 0 & 0 & \mathbf{1} & 0 & 0 \\ 0 & \mathbf{1} & 0 & 0 & 0 \\ \mathbf{1} & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & \mathbf{1} \end{array}\right] &
\left[\begin{array}{ccccc} 0 & 0 & 0 & 0 & \mathbf{1} \\ 0 & 0 & 0 & \mathbf{1} & 0 \\ 0 & 0 & \mathbf{1} & 0 & 0 \\ 0 & \mathbf{1} & 0 & 0 & 0 \\ \mathbf{1} & 0 & 0 & 0 & 0 \end{array}\right] \\
\text{lag 2} &
\left[\begin{array}{ccccc} 0 & 0 & 0 & 0 & \mathbf{1} \\ 0 & 0 & 0 & \mathbf{1} & 0 \\ 0 & 0 & \mathbf{1} & 0 & 0 \\ 0 & \mathbf{1} & 0 & 0 & 0 \\ \mathbf{1} & 0 & 0 & 0 & 0 \end{array}\right] &
\left[\begin{array}{ccccc} \mathbf{1} & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & \mathbf{1} \\ 0 & 0 & 0 & \mathbf{1} & 0 \\ 0 & 0 & \mathbf{1} & 0 & 0 \\ 0 & \mathbf{1} & 0 & 0 & 0 \end{array}\right] &
\left[\begin{array}{ccccc} 0 & \mathbf{1} & 0 & 0 & 0 \\ \mathbf{1} & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & \mathbf{1} \\ 0 & 0 & 0 & \mathbf{1} & 0 \\ 0 & 0 & \mathbf{1} & 0 & 0 \end{array}\right] &
\left[\begin{array}{ccccc} 0 & 0 & \mathbf{1} & 0 & 0 \\ 0 & \mathbf{1} & 0 & 0 & 0 \\ \mathbf{1} & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & \mathbf{1} & 0 \\ 0 & 0 & 0 & 0 & \mathbf{1} \end{array}\right] &
\left[\begin{array}{ccccc} 0 & 0 & 0 & \mathbf{1} & 0 \\ 0 & 0 & \mathbf{1} & 0 & 0 \\ 0 & \mathbf{1} & 0 & 0 & 0 \\ \mathbf{1} & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & \mathbf{1} \end{array}\right] \\
\text{lag 3} &
\left[\begin{array}{ccccc} 0 & 0 & 0 & \mathbf{1} & 0 \\ 0 & 0 & \mathbf{1} & 0 & 0 \\ 0 & \mathbf{1} & 0 & 0 & 0 \\ \mathbf{1} & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & \mathbf{1} \end{array}\right] &
\left[\begin{array}{ccccc} 0 & 0 & 0 & 0 & \mathbf{1} \\ 0 & 0 & 0 & \mathbf{1} & 0 \\ 0 & 0 & \mathbf{1} & 0 & 0 \\ 0 & \mathbf{1} & 0 & 0 & 0 \\ \mathbf{1} & 0 & 0 & 0 & 0 \end{array}\right] &
\left[\begin{array}{ccccc} \mathbf{1} & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & \mathbf{1} \\ 0 & 0 & 0 & \mathbf{1} & 0 \\ 0 & 0 & \mathbf{1} & 0 & 0 \\ 0 & \mathbf{1} & 0 & 0 & 0 \end{array}\right] &
\left[\begin{array}{ccccc} 0 & \mathbf{1} & 0 & 0 & 0 \\ \mathbf{1} & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & \mathbf{1} \\ 0 & 0 & 0 & \mathbf{1} & 0 \\ 0 & 0 & \mathbf{1} & 0 & 0 \end{array}\right] &
\left[\begin{array}{ccccc} 0 & 0 & \mathbf{1} & 0 & 0 \\ 0 & \mathbf{1} & 0 & 0 & 0 \\ \mathbf{1} & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & \mathbf{1} & 0 \\ 0 & 0 & 0 & 0 & \mathbf{1} \end{array}\right] \\
\text{lag 4} &
\left[\begin{array}{ccccc} 0 & 0 & \mathbf{1} & 0 & 0 \\ 0 & \mathbf{1} & 0 & 0 & 0 \\ \mathbf{1} & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & \mathbf{1} \\ 0 & 0 & 0 & \mathbf{1} & 0 \end{array}\right] &
\left[\begin{array}{ccccc} 0 & 0 & 0 & \mathbf{1} & 0 \\ 0 & 0 & \mathbf{1} & 0 & 0 \\ 0 & \mathbf{1} & 0 & 0 & 0 \\ \mathbf{1} & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & \mathbf{1} \end{array}\right] &
\left[\begin{array}{ccccc} 0 & 0 & 0 & 0 & \mathbf{1} \\ 0 & 0 & 0 & \mathbf{1} & 0 \\ 0 & 0 & \mathbf{1} & 0 & 0 \\ 0 & \mathbf{1} & 0 & 0 & 0 \\ \mathbf{1} & 0 & 0 & 0 & 0 \end{array}\right] &
\left[\begin{array}{ccccc} \mathbf{1} & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & \mathbf{1} \\ 0 & 0 & 0 & \mathbf{1} & 0 \\ 0 & 0 & \mathbf{1} & 0 & 0 \\ 0 & \mathbf{1} & 0 & 0 & 0 \end{array}\right] &
\left[\begin{array}{ccccc} 0 & \mathbf{1} & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & \mathbf{1} \\ 0 & 0 & 0 & \mathbf{1} & 0 \\ 0 & 0 & \mathbf{1} & 0 & 0 \\ \mathbf{1} & 0 & 0 & 0 & 0 \end{array}\right] \\
\text{lag 5} &
\left[\begin{array}{ccccc} 0 & \mathbf{1} & 0 & 0 & 0 \\ \mathbf{1} & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & \mathbf{1} \\ 0 & 0 & 0 & \mathbf{1} & 0 \\ 0 & 0 & \mathbf{1} & 0 & 0 \end{array}\right] &
\left[\begin{array}{ccccc} 0 & 0 & \mathbf{1} & 0 & 0 \\ 0 & \mathbf{1} & 0 & 0 & 0 \\ \mathbf{1} & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & \mathbf{1} & 0 \\ 0 & 0 & 0 & 0 & \mathbf{1} \end{array}\right] &
\left[\begin{array}{ccccc} 0 & 0 & 0 & \mathbf{1} & 0 \\ 0 & 0 & \mathbf{1} & 0 & 0 \\ 0 & \mathbf{1} & 0 & 0 & 0 \\ \mathbf{1} & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & \mathbf{1} \end{array}\right] &
\left[\begin{array}{ccccc} 0 & 0 & 0 & 0 & \mathbf{1} \\ 0 & 0 & 0 & \mathbf{1} & 0 \\ 0 & 0 & \mathbf{1} & 0 & 0 \\ 0 & \mathbf{1} & 0 & 0 & 0 \\ \mathbf{1} & 0 & 0 & 0 & 0 \end{array}\right] &
\left[\begin{array}{ccccc} \mathbf{1} & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & \mathbf{1} \\ 0 & 0 & 0 & \mathbf{1} & 0 \\ 0 & 0 & \mathbf{1} & 0 & 0 \\ 0 & \mathbf{1} & 0 & 0 & 0 \end{array}\right]
\end{array}
$$

Vi kan altså se, at den samme teori tilsyneladende lader sig overføre til latinske kuber. Det er også muligt, at foretage ± 1-skridt i denne forekomstmatrix, hvis blot man vælger ettaller som før i rækken, søjlen og lagene, men nu også i kuberne, så skridtene foregår i en $2 \times 2 \times 2 \times 2$ matrix i stedet for i en $2 \times 2 \times 2$ matrix. Desværre er det ikke umiddelbart muligt, at begrænse antallet af genererede -1-pladser så let som før. Da vi betragtede en $n \times n \times n$ forekomstmatrix var der én ubekendt i den udvalgte $2 \times 2 \times 2$ matrix. Denne befandt sig på en plads, hvor vi skulle trække fra, og kunne derfor give én -1-plads. Når vi nu udvælger en $2 \times 2 \times 2 \times 2$ matrix har vi 5 ubekendte, hvoraf 4 befinder sig på pladser, hvor der bliver trukket fra. Det vil altså sige, at vi kan generere op til 4 -1-pladser. På grund af indbyrdes afhængighedsforhold er der 17 forskellige måder at udfylde de 5 ukendte pladser. Heraf generere 5 situationer 0 eller 1 -1-plads og de resterende 12 situationer genererer flere. Dette betyder i praksis, at for hver gang vi fjerner en -1-plads genererer vi efter al sandsynlighed mere end en ny -1-plads. Dette betyder, at antallet af -1-pladser stiger, og det er derfor usandsynligt at ende med en ny egentlig forekomstmatrix ved tilfældigt valg. Vi ønsker ligesom før, at opstille nogle begrænsninger, som kan garantere, at vi ikke ender med en stadig voksende strøm af -1-pladser, men dette ligger udenfor rapportens fokus.

Litteratur

[1] Thomas H. Cormen, Clifford Stein, Roanls L. Rivest og Charles E. Leiserson. *Introduction to Algorithms*. The MIT Press, 2009.

[2] David Cox, John Little og Donal O'Shea. *Ideals, Varieties and Algorithms*. Springer, 2012.

[3] Reinhard Dielstel. *Graph Theory*. Springer, 2010.

[4] R. Dougherty, C. Freiling og K. Zeger. Insufficiency of linear coding in network information flow. *Information Theory, IEEE Transactions on*, 51(8):2745-2759, 2005.

[5] Louise Foshammer og Malte Neve-Græsbøll. Netværkskodning. 2013.

[6] Olav Geil, Louise Foshammer og Malte Neve-Græsbøll. Combining subspace codes with classical linear error-correcting codes. 2014.

[7] Olav Geil og Casper Thomsen. *Aspects of random network coding*, bind 8 af *Series on Coding Theory and Cryptology*, side 47-81. World Scientific Publishing Co Pte Ltd, 2013. 2013; 1.

[8] Tracey Ho, M. Medard, R. Koetter, D.R. Karger, M. Effros, Jun Shi og B. Leong. A random linear network coding approach to multicast. *Information Theory, IEEE Transactions on*, 52(10):4413-4430, 2006.

[9] Mark T. Jacobson og Peter Matthews. Generating uniformly distributed random latin squares. *Journal of Combinatorial Designs*, 4(6):405-437, 1996.

[10] S. Jaggi, P. Sanders, P.A. Chou, M. Effros, S. Egner, K. Jain og L. M G M Tolhuizen. Polynomial time algorithms for multicast network code

construction. *Information Theory, IEEE Transactions on*, 51(6):1973-1982, 2005.

[11] Jørn Justesen og Tom Høholdt. *A Course In Errorcorrection Codes*. European Mathematical Society, 2004.

[12] Ralf Kötter og Frank R. Kschischang. Coding for errors and erasures in random network coding. *IEEE Transactions on Information Theory*, 54(8):3579-3591, 2008.

[13] Ralf Kötter og Muriel Médard. An algebraic approch to network coding. *IEEE Transactions on Information Theory*, 11(5):782-795, 2003.