NTNU
Det skapende universitet

Optimale algoritmer for dynamisk frekvensbruk i DSL systemer

Vegar Hansen

Master i elektronikk
Oppgaven levert: Juni 2008
Hovedveileder: Nils Holte, IET

Norges teknisk-naturvitenskapelige universitet
Institutt for elektronikk og telekommunikasjon

Oppgavetekst

I dagens standarder for DSL (Digital Subscriber Lines) er det satt maksimalverdier for utsendt effekttetthet i ulike frekvensområder. Kompatibiliteten mellom DSL systemene ivaretas ved at alle brukere ligger innenfor disse maksimalverdier. Dette gjør at dimensjonering av systemene må baseres på verste tilfelle av interferens, noe som fører til relativt pessimistiske estimater for rekkevidder og bitrater.

Det arbeides internasjonalt med å ta i bruk en dynamisk frekvensbruk i DSL systemer (DSM - Dynamic Spectral Management) for å oppnå økt ytelse. Denne oppgaven skal ta for seg optimale algoritmer for DSM hvor det forutsettes en sentralisert koordinering av alle DSL systemene i samme kabel.

Oppgaven gitt: 15. januar 2008
Hovedveileder: Nils Holte, IET

Sammendrag

Denne masteroppgaven er en fortsettelse av en prosjektoppgave gjennomført høsten 2007. Den går ut på å sette seg inn i problemet med krysstale i DSL (Digital Subscriber Lines), og da spesielt en av de beste løsningene på dette problemet, dynamisk frekvensbruk (DSM – Dynamic Spectral Management). Dette kan løses på tre nivåer, med økende kompleksitet og økende datarater ved høyere implementering. Nivå 1 omfatter distribuerte systemer, der hver bruker justerer sitt eget effektspektrum for å oppnå ønskede rater. I nivå 2 er det en sentralisert kontroller som beregner spektra for flere brukere over samme kabel. Nivå 3 trenger også en kontroller, men her blir hele kanalen undersøkt for å finne kanalmatrisen. Denne kan benyttes til å fjerne og i beste fall utnytte krysstalen.

I prosjektoppgaven høsten 2007 ble en av de enkleste formene for DSM sammenlignet med optimale løsninger for krysstalekoordinering (OSM – Optimal Spectrum Management). OSM er den beste løsningen for DSM uten å ta i bruk krysstalekansellering. Søket for å finne den optimale løsningen ble gjort ved hjelp av Exhaustive search, som går igjennom alle mulige kombinasjoner og finner den beste. Konklusjonen var da at store forbedringer, spesielt med OSM, var mulige med innføring av DSM. Ulempen med Exhaustive search er at den blir veldig kompleks med økende antall delbånd og brukere, $O(e^{KN})$. Derfor ble kun to brukere og tre delbånd benyttet. Dette klarte allikevel å vise hvilken effekt OSM har.

Siden Exhaustive search var så kompleks ble det vurdert at andre algoritmer eventuelt måtte benyttes for implementering. Masteroppgaven tok derfor for seg Dual decomposition. Denne avanserte algortimen optimaliserer for hvert delbånd i stedet for på tvers av alle PSD. Fordelen med dette er at kompleksiteten blir vesentlig mindre fordi den blir lineær med hensyn på antall delbånd som blir benyttet, og ikke eksponentiell. På grunn av innføringen av Lagrangemultiplikatorer vil kompleksiteten stige noe på grunn av dette, men det er uvesentlig sammenlignet med Exhaustive search. Den eneste støyen som eksisterte i systemet var hvit støy og krysstale fra trafikk med samme retning (FEXT – Far End Crosstalk).

Det ble vist at beregningstidene for Dual decomposition var vesentlig mye mindre for økende bitloading og delbånd enn for Exhaustive search. Med Dual decomposition var det mulig å øke både bitloading og delbånd for å oppnå store rateregioner innen relativt kort tid. Da rateregionene ble sammenlignet var det mulig å observere at Dual decomposition ikke alltid klarte å finne absolutt alle optimale punkter. Dette vil gjerne være uvesentlig tatt i betraktning at den er såpass mye mer effektiv.

Ulempen med Dual decomposition er at kompleksiteten fremdeles er eksponentiell med hensyn på antall brukere. Derfor vil også Dual decomposition bli vanskelig å innføre i de systemene som blir benyttet av mange brukere. Fremtidig arbeid vil derfor være å se på andre metoder som er mer lineære med hensyn på antall brukere.

Forord

Denne masteroppgaven er gjennomført på Norges teknisk-naturvitenskapelige universitet (NTNU) våren 2008, som en avslutning på Masterstudiet i Elektronikk. Høsten 2007 hadde jeg en prosjektoppgave som omhandlet samme tema, bare med mindre kompliserte algoritmer. Det har i løpet av denne tiden dukket opp en del problemer, men ved hjelp av veileder og annen støtte syns jeg nå det er et produkt jeg kan stå inne for. Dette sammen med andre faktorer har ført til at arbeidet med oppgaven har vært en veldig lærerik prosess.

Jeg vil med dette takke hovedveilederen min, professor Nils Holte, for å ha tilføyd sin visdom samt veiledet meg gjennom nesten hele oppgaven. En stor takk må også rettes til min andre veileder, professor II Terje Røste, som tok over mot slutten av oppgaven og hjalp meg i havn. Andre som må takkes for støtte og hjelp er Camilla, familien min og vennene mine.

Trondheim, 10.06.08
Vegar Hansen

Innhold

Forkortelser og matematiske symboler

Forkortelser

ACS – Auto-Configuration Server
ADSL – Asymmetric DSL
CAP – Carrierless Amplitude Phase modulation
CPE – Customer Premise Equipment
CPU – Central Processing Unit
DMT – Discrete Multi-Tone
DS – Downstream
DSL – Digital Subscriber Line
DSLAM – DSL Access Multiplexer
DSM – Dynamic Spectrum Management
DSM-C – DSM Control interface
DSM-D – DSM Data interface
EMS – Element Management System
FDD – Frequency Division Duplexing
FEC – Forward Error Correction
FEXT – Far End Crosstalk
HDTV – High Definition Television
ICI – Inter Carrier Interference
ISDN – Integrated Services Digital Network
LT – Line Terminal
MIMO – Multiple Input Multiple Output
NEXT – Near End Crosstalk
NT – Network Terminal
OSM – Optimal Spectrum Management
PSD – Power Spectral Density
QAM – Quadrature Amplitude Modulation
RAM – Random Access Memory
SMC – Spectrum Maintenance Centre
SSM – Static Spectrum Management
US – Upstream
VDSL – Very high speed DSL

Matematiske symboler

$a_{k,n}$ – Kanaldempingen for bruker n og tone k
$b_{k,n}$ – Bitloadingrate på tone k for bruker n
b_{max} – Maksimal støttet bitloading av modem
$p_{k,n,m}$ – Det inverse av krysstaledempningen fra bruker m på bruker n i tone k
P_n – Maksimal effekt sendt for bruker n
f_s – DMT symbolrate
K – Antall delbånd/toner
L – Lagrangefunksjonen
L_k – Lagrangefunksjonen på tone k
l_n – Lengden på kabelen til bruker n
N – Antall brukere
q_k – Mulige PSD kombinasjoner for tone k
R_n – Dataraten til bruker n
$s_{k,n}$ – PSD på tone k for bruker n
w – Vektingen i Dual decomposition
Γ – Margin gitt av Shannon
Δ_f – Frekvensbåndinndelingen
ε – Nøyaktigheten på intervallhalveringsmetoden (bisection)
$\sigma_{k,n}$ – Gaussisk kanalstøy på bruker n og tone k
λ_n – Lagrangemultiplikatoren til bruker n

Innledning

Den suksessfulle installeringen av store fiberbaserte nettverk over hele verden har ført til økte datarater. Dette er en trend som ser ut til å fortsette. Mens høye datarater kan oppnås til hver enkelt bruker over fiber, er DSL en mye billigere metode å fullføre denne siste delen inn til brukeren på. Dette fordi DSL benytter tvunnet kabel, noe som allerede ligger inn til så godt som alle brukerne.

Det økte behovet for økt kapasitet har ført til stor utvikling innen DSL teknologien. Dette på grunn av at kapasiteten er avhengig av lengden på kopperlinjene, diameteren til de tvinnede parene, grad av symmetrisk og asymmetrisk trafikk, kodingen, osv. I det siste har det vært mer og mer fokus på å ta i bruk dynamisk frekvensbruk i DSL for å motvirke krysstalen som oppstår i kablene. Dette kalles DSM (Dynamic Spectrum Management) og kan love større forbedringer i kapasiteten enn noe annet.

Forskningen innenfor dette området har vært og er stor. Innenfor de to formene som ikke tar i bruk krysstalekansellering er det produsert optimale algoritmer for å finne kapasiteten. Problemet med disse er at de fremdeles er noe komplekse. Å finne forenklinger til dette er det hovedtyngden av forskningen i dag går ut på.

Som oppgaveteksten sier skal denne oppgaven ta for seg optimale algoritmer for DSM. Det forutsettes en sentralisert koordinering av alle DSL systemene i samme kabel, også kalt DSM nivå 2. Dette skal gjøres ved hjelp av simuleringer. En av algoritmene som skal vurderes er en algoritme som leter igjennom alle mulige kombinasjoner av effektfordeling, også kalt Exhaustive search. Dette er en algoritme som blir umulig å bruke i praksis når antall delbånd økes. Derfor skal denne sammenlignes med en annen algoritme, kalt Dual decomposition [6], som optimaliserer ved hjelp av Lagrange på hvert delbånd. Denne er mer aktuell fordi kompleksiteten øker lineært, i stedet for eksponentielt som Exhaustive search gjør, for økning av antall delbånd. Med disse algoritmene kan optimal løsning finnes. Dette er nyttig for å vite hvor bra det kan bli uten å ta i bruk krysstalekansellering.

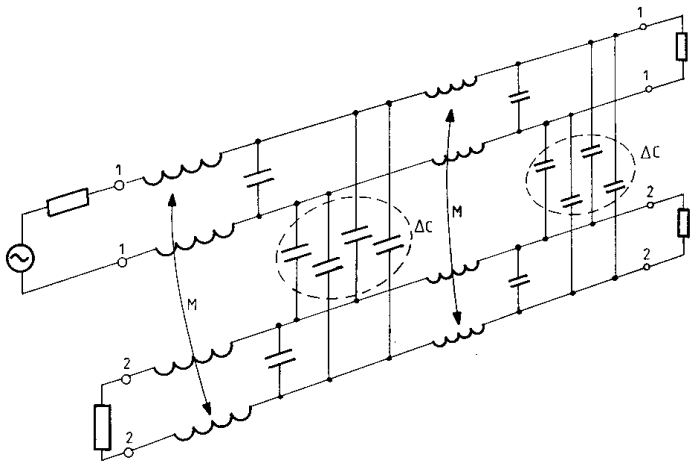
Rapporten tar først for seg den teorien som er nødvendig for å forstå problemet og å klare å gjennomføre simuleringene i kapittel 1 Teori. Her diskuteres kopperkablene som brukes til tvinnede par (for å se hva som er problemet med kopperkabler), DSM på alle nivå og VDSL (siden det er VDSL som skal benyttes i simuleringene). Det er spesielt DSM nivå 2 med størst vekt på Dual decomposition som diskuteres, da det er dette som skal analyseres senere. I kapittel 2 Simuleringer / Resultater blir systemet som skal simuleres introdusert og simulert. Deretter blir resultatene diskutert og sammenlignet. Kapittel 3 Konklusjon inneholder konklusjonen og et lite avsnitt om eventuelt videre arbeid innenfor området.

1 Teori

1.1 Kopperkabler

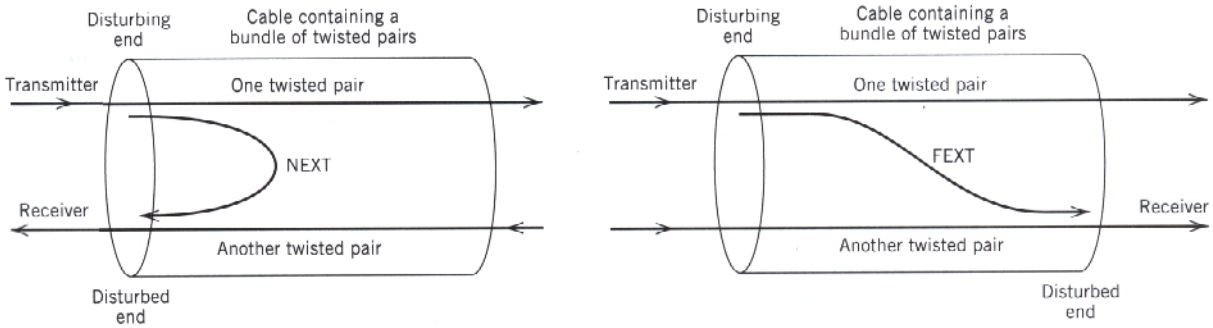
Kopperkabler har fra eldre tider kun vært brukt til offentlig telekommunikasjon, der kun de fire laveste kilohertzene har blitt brukt. Med innføringen av bredbånd til allmennheten, ble forskjellige xDSL (Digital Subscriber Line) systemer laget ved hjelp av avansert signalprosessering for å utnytte at en enkelt kopperkabel minst har 10 MHz båndbredde.

Bortsett fra hvit støy er problemet med kopperkabler at når det ligger flere kabler ved siden av hverandre (gjerne opp mot 2000 i grupper på 10 til 50 par) vil det oppstå elektromagnetisk interferens mellom alle parene i kabelen, noe som kalles krysstale. Krysstalen fra de nærliggende parene ligger gjerne 10 til 20 dB over hvit støy. Koblingen består av induktiv og kapasitiv kobling, illustrert i Figur 1.1.1, som viser koblingen mellom to par. I praksis er denne koblingen stokastisk fra par til par og langs hele kabelen.



Figur 1.1.1 Krysstalekobling mellom to par [1]

Det er to typer krysstale, NEXT (Near End Crosstalk), som er interferens mellom motsatt retning motsatt retning i forhold til overføringsretningen, og FEXT (Far End Crosstalk), som er interferens mellom systemer som bruker samme overføringsretning. Figur 2.1.2 illustrerer dette.



Figur 1.1.2 NEXT og FEXT – illustrasjon [2]

Siden FEXT, i motsetning til NEXT, blir dempet langs kabellengden, vil NEXT i de fleste tilfeller dominere over FEXT. En relativt enkel måte å unngå NEXT på er ved å benytte atskilte bånd for opp-, og nedstrøms trafikk, noe som lønner seg for høyere frekvens

dataoverføring. Dette blir utnyttet i blant annet ADSL og VDSL, mens blant annet ISDN benytter to-veis transmisjon.

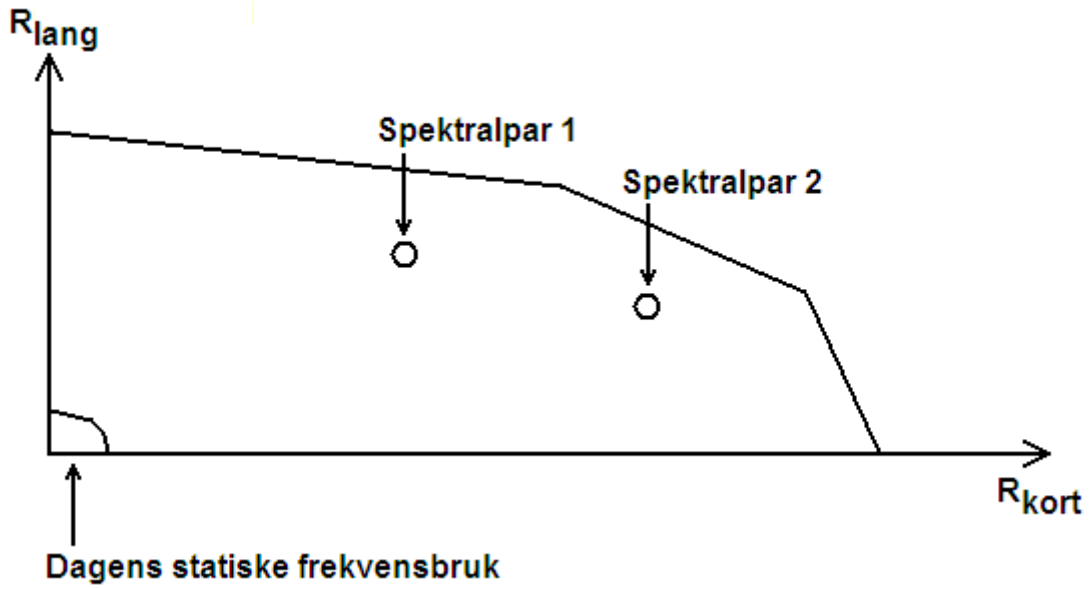
Uansett om man utelukker NEXT eller ikke, vil det alltid være krysstale mellom kablene som må taes med i beregningene av mulige datarater. Det som har blitt brukt frem til i dag er identiske maksimalverdier for utsendt effekt (også kalt SSM (Static Spectrum Mangement)) i alle modemene. Dette er ikke en veldig effektiv måte å løse problemet på siden man antar verste tilfelle til enhver tid. For å oppnå økt ytelse kan dette løses med å ta i bruk dynamisk frekvensbruk i DSL systemer (DSM – Dynamic Spectral Management) som tilpasser seg til forskjellige (gjerne varierende) systemer.

1.2 DSM (Dynamic Spectrum Management)

Å forklare DSM blir sagt i [5] på denne måten: Å ikke ha noen form for spektrumskontroll er som å kjøre i et land uten noen trafikklover, uten trafikksignaler og uten politi. Statisk spektrumskontroll, det som blir brukt for det meste i dag, er som å kjøre med trafikklover, signaler og politi kontrollert av et ultrakonservativt diktatur, der sjåfører noen ganger blir tvunget av loven inn i kollisjoner. DSM (Dynamic Spectrum Management) er som å ha gjensidig fordelaktig lover med trafikkontrollerte stopplys, flere kjørefiler og sjåfører trent til å respektere kjørefilene og å unngå kollisjoner.

DSM er den nyeste av DSL metodene som muliggjør en høyst ønskelig forbedring til DSL, der noen av fordelene er

- automatisk deteksjon og/eller forbygging av tjenestefeil (dvs. lavere rater enn antatt) på grunn av krysstale
- bedre miks av symmetrisk og usymmetriske tjenester
- høyere og mer pålitelige datarater



Figur 1.2.1 Rateregion for to brukere [4]

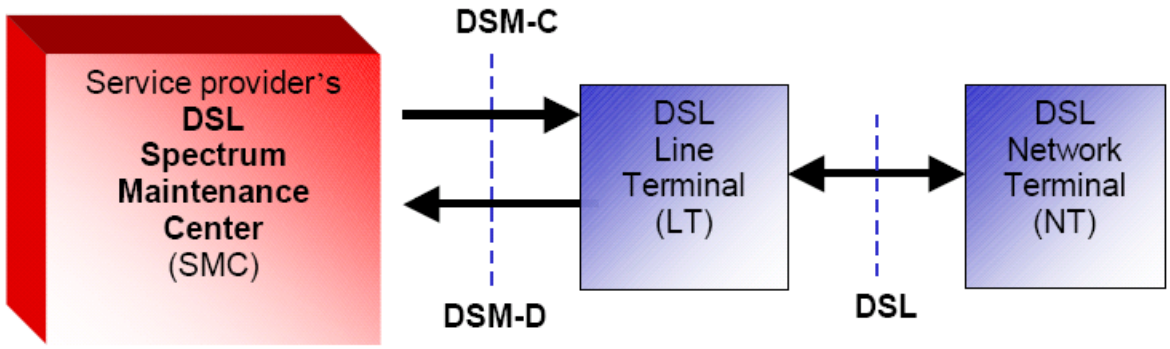
Figur 1.2.1 illustrerer konseptet av en rateregion i spektrumsbalansering. To gjensidig krysstalende DSL tjenester deler en kabel, der hver virker som støy på den andre. Den korte linjen har gjerne høyere datarater og kan lage vesentlig støy på den lengre linjen, spesielt når den korte linjen sin mottaker er nære den lange linjens mottaker på sentralsiden (oppstrøms). Hvert eneste punkt i rateregionen er oppnåelige med forskjellige spektra for brukerne. Dette

medfører mange forskjellige DSL hastigheter for begge brukerne. Hvor mange er avhengig av kontinuerlig eller diskret bitloading (antall bit/s/Hz tildelt per tone), der kontinuerlig bitloading betyr at modemene kan motta alle mulige bitloadinger. Siden praktiske DSL modem kun støtter et fast sett med diskrete bitloadinger, vil dette ofte være mest aktuelt.

DSM kan sees på som en adaptiv form for kontrollering av spektrumet. DSM har tre nivå (4 hvis man tar med SSM som nivå 0), der økende DSM nivå kan sees på som en utvikling mot økende koordinering mellom flere interfererende DSL linjer. Fra nivå 1 og 2 (flerbruker effekttildeling for å unngå krysstale) til nivå 3 (flerbruker deteksjon for å dempe krysstalen). Nivå 1 består av distribuerte algoritmer, der hvert system kun optimalsiserer sin egen overføring uten koordinering med de andre autonome systemene. I nivå 2 kan løsningen bli så optimal som mulig uten krysstalekansellering med en sentralisert kontroller, også kalt SMC (Spectrum Maintenance Centre), som koordinerer systemene. Nivå 3 omfatter fullstendig krysstalekansellering med å konstruere en vektor for hele kanalen. Dette blir det som kalles MIMO (Multiple Input, Multiple Output) i kabel for å levere optimalt datagjennomstrømming, gjerne ved å utnytte krysstalen. En oversikt over nivåene er gitt i Figur 1.2.2.

DSM nivå	Funksjonalitet
Nivå 0	SSM (Static Spectrum Management)
Nivå 1	Singel par effekttildeling for å motvirke krysstale
Nivå 2	Flere par koordinert effekttildeling for å motvirke krysstale
Nivå 3	Flere par, MIMO (Multiple Input, Multiple Output) for å motvirke (og eventuelt utnytte) krysstale

Figur 1.2.2 DSM nivåer [5]



Figur 1.2.3 DSM referansediagram [5]

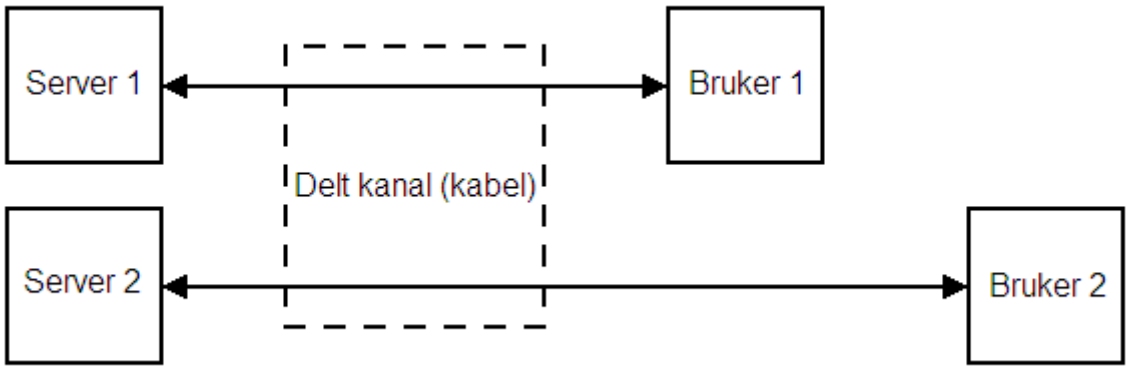
I Figur 1.2.3 kan man se et grunnleggende referansediagram for DSM. Her er SMC (Spectrum Maintenance Centre) serveren som brukerne vil koble seg opp til for å kommunisere videre. Mellom linjeterminalen og brukeren/nettverksterminalen (NT) er det standard DSL-interface som benyttes. DSM-D grensesnittet transporterer data mens DSM-C grensesnittet (også kalt kontrollgrensesnittet) transporterer kommandoer vanligvis kjent som ”profilinnstillinger” (”profile settings”) til DSL linjene. Dette er grundigere forklart i [5].

For denne oppgaven er det to brukere, altså to nettverksterminaler, som skal koble seg opp til serveren via en felles linjeterminal. Ut i fra om det er datatrafikk som skal gå opp til serveren (oppstrøm) eller ned til brukerne (nedstrøm) vil det bli forskjellige rateregioner. I denne oppgaven er det oppstrømstrafikk som skal undersøkes.

1.3 DSM: Nivå 1 – Distribuert

I DSM nivå 1 vil hvert av systemene altså justere seg selv. Dette skjer uten noen koordinering systemene imellom. Allikevel, på grunn av sin relativt lave kompleksitet, blir nivå 1 testet ut for flere DSL nettverk i dag. Nivå 1 tar nytte av data- og kontrolleringsinterferens for å monitorere spektrumsnivåene og grad av forover feilkorrigering (FEC) på individuelle brukere. Hver bruker har altså ingen informasjon om andre brukere. Dette benyttes til justering når det er hensiktsmessig. Slike system kan sees på som en del av SMCen administrert av serviceleverandøren. Eksempel på nivå 1 DSM kan finnes i [4].

Hvordan blokkdiagrammet for nivå 1 DSM ser ut for to brukere blir noe forenklet gjengitt i Figur 1.3.1. Dette blokkdiagrammet gjelder også for nivå 0 (SSM) siden det ikke er noen koordinering for noen av tilfellene. Forskjellen er innad i hvert system/bruker, der nivå 1 bruker algoritmer for å forandre PSD.



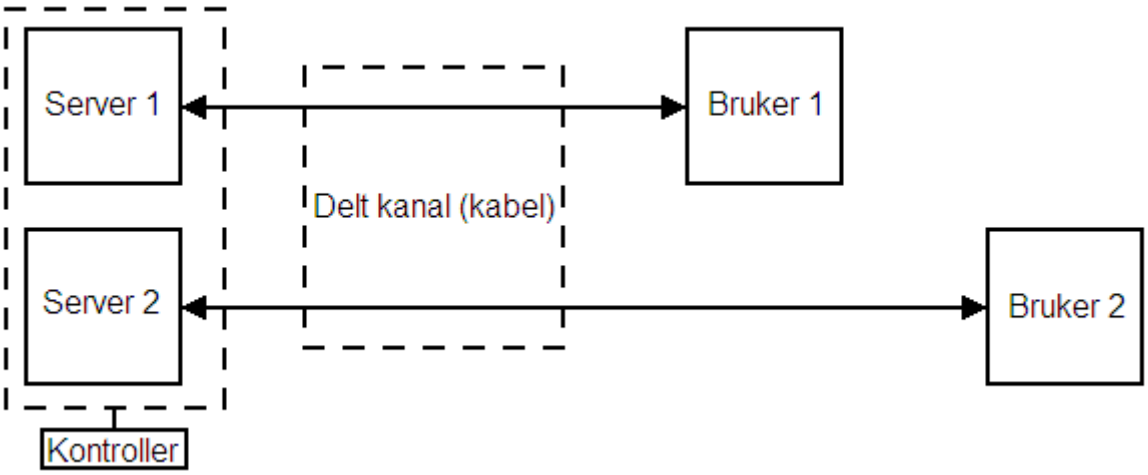
Figur 1.3.1 Autonome system uten koordinering

En enkel utgave av nivå 1 DSM er at hver bruker måler og finner adaptivt ut hvilken effekt den minimum må sende for å oppnå en ønsket rate. En annen metode som blir jobbet med i disse dager er iterativ waterfilling, som kort fortalt går ut på å ”sende mest når kanalen er god”. Det er denne metoden som kan brukes til å finne den maksimalt oppnåelige dataraten for DSM i sin enkleste implementering.

Denne oppgaven skal ikke behandle nivå 1. Dersom det er ønskelig med utfyllende informasjon om dette, anbefales det å lese annen litteratur.

1.4 DSM: Nivå 2 – Sentralisert

Nivå 2 systemer tar i bruk kabelkarakterisering. Dette betyr i hovedsak kjennskapen til kabelinformasjon (lengde, tykkelse, osv) eller krysstalekoblingen mellom noen eller flere par i kabelen. Sentralisert ”spectrum management” kan benyttes i de tilfellene en sentralisert kontroller (SMC) er tilgjengelig. Denne er ansvarlig for å beregne effektspektra til alle brukerne. For å sikre den optimale ytelsen med sentraliserte algoritmer er det nødvendig med reoptimalisering hver gang en bruker kobler til eller fra. Dette er en av ulempene med sentraliserte algoritmer sammenlignet med distribuerte algoritmer. En annen ting er at sentraliserte algoritmer behøver en kontroller som ikke finnes i ubundet nettverk hvor for eksempel flere leverandører deler den samme kabelen. I slike tilfeller vil gjerne distribuerte algoritmer være å foretrekke. Det som kan være nyttig i en slik sammenheng er en maksimumsgrense for hvor bra det kan gjøres, noe som diskuteres i 1.4.1. Denne maksimumsgrensen kan brukes for både distribuerte og sentraliserte algoritmer, for å ha et ønsket oppnådd mål. Et illustrerende blokkdiagram for nivå 2 kan man se i Figur 1.4.1.



Figur 1.4.1 Optimal løsning uten krysstalekansellering

1.4.1 OSM (Optimal Spectrum Management)

OSM (Optimal Spectrum Mangement) tilhører DSM nivå 2 og beregner den teoretiske optimale PSD (Power Spectral Density) for alle modem i DSL systemer. Måten å gjøre dette på er å bruke optimaliseringsteori.

DMT (Discrete MultiTone) modulasjon har blitt brukt mer og mer for forskjellige xDSL applikasjoner og er i dag standard for ADSL og VDSL. Hvis modemene innen ett nettverk er synkronisert fører dette til at ICI (Inter Carrier Interference) så godt som forsvinner. Under antagelsen om at man har en gaussisk kanal ut fra Shannon med margin Γ blir den oppnådde bitloadingraten $b_{k,n}$ for bruker n på tone k gitt i Formel 1.4.1. Maksimal støttet bitloading b_{max} av modemene ligger innenfor 8-15 i de fleste sammenhenger i dag. Det er også antatt at modemene behandler interferens fra de andre brukerne som støy.

$$b_{k,n} = \min\left(b_{\max}, \log_2\left(1 + \frac{1}{\Gamma} \frac{a_{k,n} s_{k,n}}{\sum_{m \neq n} p_{k,n,m} s_{k,m} + \sigma_{k,n}} \right) \right)$$

Formel 1.4.1

I Formel 1.4.1 er $s_{k,n}$ PSD på tone k for bruker n, $a_{k,n}$ kanaldempingen for bruker n og tone k, $p_{k,n,m}$ er det inverse av krysstaledempningen fra bruker m på bruker n i tone k. $\sigma_{k,n}$ er gaussisk kanalstøy på bruker n og tone k. Total rate blir for bruker n blir dermed $R_n = f_s \sum_k b_{k,n}$, der f_s er DMT symbolrate.

Målet med "spectrum management" er å oppnå en bedre tradeoff mellom alle brukerne i ett nettverk. For enkelhetens skyld sees det på et tobrukersystem. Dermed maksimeres raten til bruker 2, R_2, mens raten til bruker 1 er over minimum rate, $R_{1,target}$. Anta at hver enkelt bruker har en maksimal effekt de kan bruke, P_n. Matematisk formulering av dette problemet er gitt i Figur 1.4.2. Δ_f er bredden på båndene for tonene k.

Raten til bruker 2 skal maksimeres, gitt at:	$\max_{s_1,s_2} R_2$
- Raten til bruker 1 må være over en minimal verdi	s.t. $R_1 \geq R_{1,target}$
- Effekten til bruker n må ikke overgå maksimal verdi	s.t. $\Delta_f \sum_k s_{k,n} \leq P_n, n = 1,2$
- Effekten brukt på hver tone må være over null og, om det er ønsket, ikke overgå en maksimal verdi	$0 \leq s_{k,n} \leq s_{k,n,\max}$

Figur 1.4.2 Spectrum Management Problem [5][6]

Avhengig av om det brukes kontinuerlig eller diskret bitloading vil antall PSD kombinasjoner variere. Anta kontinuerlig bitloading med oppløsning Δ_s i modemene. I dagens standarder er Δ_s normalt satt til 0,5 dBm/Hz. Med positiv sendeeffekt med maksimalt $s_{k,n,max}$ vil de $s_{k,n}$ som er aktuelle bli $s_{k,n} \in \{0, \Delta_s, ..., s_{k,n,\max}\}$. Dette vil medføre $q_k = \prod_n (s_{k,n,\max} \Delta_s^{-1} + 1)$ mulige PSD kombinasjoner for tone k. Med diskret bitloading kan aktuelle $s_{k,n}$ beregnes ut i fra Formel 1.4.1 noe som for et tobrukertilfelle vil medføre en utvalg av PSD kombinasjoner på $(s_{k,1}, s_{k,2}) \in \{(s_{k,1}(b_{k,1}, b_{k,2}), s_{k,2}(b_{k,1}, b_{k,2})) \mid b_{k,n} \in \{0,...,b_{\max}\} \forall n\}$ noe som vil medføre $q_k = (b_{\max} + 1)^2$ mulige PSD kombinasjoner for tone k.

Løsning på dette problemet kan finnes ved å benytte ”brute force” med Exhaustive search som går igjennom aktuelle PSD for brukerne. Uheldigvis er dette en ikke-konveks optimering og behøver kompleksitet $O(e^{KN})$ for finne en løsning, der K er antall toner i systemet og N er antall brukere. For tobrukertilfellet med diskret bitloading vil kompleksiteten bli $O((b_{max} + 1)^{2K})$. En skisse av dette for bare 2 brukere, 2 toner og 2 diskrete effektnivåer (Lav eller Høy) er gitt i Figur 1.4.3. Dette enkle eksemplet gir 16 forskjellige måter å fordele effekten på.

		Bruker 1			
		LL	HL	LH	HH
Bruker 2	LL	LL / LL	HL / LL	LH / LL	HH / LL
	HL	LL / HL	HL / HL	LH / HL	HH / HL
	LH	LL / LH	HL / LH	LH / LH	HH / LH
	HH	LL / HH	HL / HH	LH / HH	HH / HH

Figur 1.4.3 OSM problem for 2 brukere, 2 toner og 2 effektnivå

Det observeres at for ADSL med K=256 og VDSL med K=4096 blir dette fort vanskelig å beregne. Dette medfører et ønske om å finne metoder som klarer å finne optimal effekttildeling uten å bruke ”brute force”. Det er flere måter å regne ut hva som er optimalt, eller tilnærmet optimalt, blant annet Dual decompostion som skal prøves ut mot Exhaustive search. Det finnes en nyere metode betegnet ”Concave minimization” [7] som skal være en mer effektiv algoritme enn ”Dual Decomposition”. Denne er ikke vurdert i denne oppgaven.

1.4.2 Dual decomposition

I [6] blir en sentralisert algoritme, basert på Dual decomposition, introdusert. Denne algoritmen løser problemet for hver tone på en effektiv og mer beregningsvennlig måte. Denne algoritmen skal vise betydningsfull ytelsesgevinst i forhold til eksisterende DSM teknikker. I [6] hevder de at algoritmen skal øke dataraten med en faktor av fire sammenlignet med den distribuerte DSM algoritmen iterativ waterfilling.

For Exhaustive search har man at totalt mulige effektrestriksjoner på hver linje assosiert med hver linjes parvise effektrestriksjoner på tvers av frekvens fører til et søk på tvers av alle toner. Dette medfører dermed en eksponentiell kompleksitet med hensyn til K og etter hvert et praktisk uberegnelig problem.

Dual decomposition metoden er en hyppig brukt fremgangsmåte i konveks optimaliseringsteori for å løse betingede optimaliseringsproblemer ved hjelp av et ekvivalent ubetinget todelt problem. Dette todelte problemet kan dekomponeres inn i flere delproblem. Dual decomposition har også blitt brukt i andre kommunikasjonsproblemer. I [6] blir det da altså vist at Dual decomposition metoden kan anvendes på ikkekonvekse optimaliseringer.

I de påfølgende avsnittene skal Dual decomposition brukes til å transformere problemet til et ekvivalent problem som er lineær med hensyn på K og dermed blir enklere å regne ut.

1.4.2.1 Vektet sum

Hvis rateregionen (se eksempel på rateregion i Figur 1.2.1) er konveks blir optimaliseringsproblemet i Figur 1.4.2 ekvivalent med vektet optimering av ratesummen gitt i Figur 1.4.4. Om rateregionen er konveks eller ikke behandles i [6] kapittel IV-A tar opp. Der blir det vist at rateregionen er tilnærmet konveks for DSL, noe som gjør at man kan finne omtrent alle operasjonspunkt.

Med en korrekt valgt w, vil maksimering av ratesummen automatisk håndheve at raten til bruker må være over et minimum. Dette relativt omfattende beviset kan finnes i [6] kapittel IV-A, der det er vist ved hjelp av en illustrasjon.

$$\max_{s_1,s_2} wR_1 + (1-w)R_2$$
$$\text{s.t. } \Delta_f \sum_k s_{k,n} \le P_n, n = 1,2$$
$$0 \le s_{k,n} \le s_{k,n,mask}$$

Figur 1.4.4 Spectrum Management Problem med vektet sum

1.4.2.2 Dual decomposition

Siden det er mulig, som vist i forrige avsnitt, å løse optimaliseringsproblemet i Figur 1.4.2 ved hjelp av vektet sum optimalisering vist i Figur 1.4.4, vil dette avsnittet vise hvordan den vektede sum av rate optimaliseringen kan bli løst på en mye enklere måte.

For forenkling av algoritmen som finner de optimale punktene løses optimaliseringsproblemet ved å først definere Lagrangefunksjonen
$L = wR_1 + (1+w)R_2 - \lambda_1 \sum_k s_{k,1} - \lambda_2 \sum_k s_{k,2}$, der λ_n er Lagrangemultiplikatoren for bruker n.

Dette medfører at maksimeringen i Figur 1.4.4 kan løses ved å løse $\max_{s_1,s_3} L(w, \lambda_1, \lambda_2, s_{k,1}, s_{k,2})$.

Hvis Lagrangefunksjonen for tone k defineres som
$L_k = wb_{k,1} + (1-w)b_{k,2} - \lambda_1 s_{k,1}(b_{k,1}, b_{k,2}) - \lambda_2 s_{k,2}(b_{k,1}, b_{k,2})$, er det lett å se at Lagrangefunksjonen for alle tonene kan dekomponeres til en sum over tonene av L_k, det vil si $L = \sum_k L_k$. Det er dette som er kjent som Dual decomposition. Resultatet av dette blir at optimeringen kan splittes inn i K per-tone optimering som er koblet sammen med w, λ_1 og λ_2. Dette vil føre til en lineær, isteden for en eksponentiell, kompleksitet for K.

1.4.2.3 Algoritmen

I Figur 1.4.5 er algoritmen som skal implementeres gjengitt. Eventuelle delbåndsbegrensninger for effekten kan implementeres ved å sette L_k til $-\infty$ hvis $s_{k,1} > s_{k,1,max}$ eller $s_{k,2} > s_{k,2,max}$. Hvis diskret bitloading skal benyttes, vil funksjonen optimize_s bli begrenset til å søke gjennom de PSD kombinasjonene som korresponderer til gyldige bitloading kombinasjoner, diskutert i 1.4.1. Om kontinuerlig bitloading benyttes, blir maksimeringen begrenset til de verdiene av $s_{k,n}$ støttet av oppløsningen til modemet, også diskutert i 1.4.1.

Det er nødvendig å søke gjennom både λ_1 og λ_2 for å finne verdier som gir tilstrekkelig betydning av effektbegrensningen i Lagrangefunksjonen gitt i 1.4.2.2. Ved å variere w blir det mulig å kartlegge de optimale punktene på det konvekse området av rateregionen.

Algoritmen består av tre løkker. En ytre løkke som varierer w, en mellomløkke som søker etter λ_1 og en indre løkke som søker etter λ_2. Bisectioning (intervallhalveringsmetoden) er brukt i hvert søk. Bisection går ut på å finne et viss punkt mellom to punkter ved å halvere intervallet ned til den nøyaktigheten du ønsker, se for øvrig [9]. Ved søk etter λ_n er det først nødvendig å finne en verdi av λ_n som sørger for at effektbegrensingen gitt av bruker n er tilfredsstilt. Denne verdien lagres i $\lambda_{n,max}$. Det er greit å legge merke til at en større λ_n legger mer vekt på effektbegrensningen for bruker n i Lagrangefunksjonen. Som et resultat, ved å bruke en større λ_n vil man oppnå en lavere total effekt for bruker n.

Når $\lambda_{n,max}$ er funnet, fortsetter algoritmen til bisection. Legg merke til at når algoritmen er ferdig er enten $\Sigma_k s_{k,n} = P_n$ eller $\lambda_n = 0$ for begge brukerne. Derfor vil Lagrangefunksjonen og det originale målet bli ekvivalent.

Det er greit å legge merke til at funksjonen optimize_s gjerne må utføre et N-dimensjonalt exhaustive search som altså er eksponentielt med hensyn på antall brukere N.

Hovedfunksjon

for $w = 0,...,1$
  $\mathbf{s}_1, \mathbf{s}_2 = optimize\_\lambda_1(w)$
end

Funksjon s_1,s_2 = optimize_ λ_1(w)

$\lambda_{1,\max} = 1$
$\lambda_{1,\min} = 0$
while $\sum_k s_{k,1} > P_1$
  $\lambda_{1,\max} = 2\lambda_{1,\max}$
  $\mathbf{s}_1, \mathbf{s}_2 = optimize\_\lambda_2(w, \lambda_{1,\max})$
end
repeat
  $\lambda_1 = (\lambda_{1,\max} + \lambda_{1,\min})/2$
  $\mathbf{s}_1, \mathbf{s}_2 = optimize\_\lambda_1(w, \lambda_1)$
  if $\sum_k s_{k,1} > P_1$
    $\lambda_{1,\min} = \lambda_1$
  else
    $\lambda_{1,\max} = \lambda_1$
  end
until convergence

Funksjon s_1,s_2 = optimize_ λ_2(w, λ_1)

$\lambda_{2,\max} = 1$
$\lambda_{2,\min} = 0$
while $\sum_k s_{k,2} > P_2$
  $\lambda_{2,\max} = 2\lambda_{2,\max}$
  $\mathbf{s}_1, \mathbf{s}_2 = optimize\_s(w, \lambda_1, \lambda_{2,\max})$
end
repeat
  $\lambda_2 = (\lambda_{2,\max} + \lambda_{2,\min})/2$
  $\mathbf{s}_1, \mathbf{s}_2 = optimize\_s(w, \lambda_1, \lambda_2)$
  if $\sum_k s_{k,2} > P_2$
    $\lambda_{2,\min} = \lambda_2$
  else
    $\lambda_{2,\max} = \lambda_2$
  end
until convergence

Funksjon s_1,s_2 = optimize_ s(w, λ_1, λ_2)

for $k = 1,...,K$
  $s_{k,1}, s_{k,2} = \arg\max_{s_{k,1}, s_{k,2}} L_k(s_{k,1}, s_{k,2}, w, \lambda_1, \lambda_2)$
  (*løses ved hjelp av et exhaustive* $2D - søk$)
end

Figur 1.4.5 Optimal Spectrum Balancing algoritme [6]

Beviset på at algoritmen i Figur 1.4.5 leverer ønskede resultater er vist i [6] sin appendix.

1.4.2.4 Kompleksitet

For Dual decomposition vil kompleksiteten bli litt mer innviklet. Dette på grunn av innføringen av Lagrangemultiplikatorene λ_1 og λ_2. Det skal i algoritmen i Figur 1.4.5 først beregnes maksimalverdier av multiplikatorene. Dette gjøres enkelt ved å doble de til effektbegrensingene er tilfredsstilt. Dette skjer typisk etter noen få iterasjoner, og vil ha ubetydelig innvirking på kompleksiteten.

Det er når man virkelig skal nærme seg effektbegrensingene at kompleksiteten øker. Da brukes bisection for å komme så nære som man ønsker. Anta at nøyaktigheten ε er ønsket for hver λ. Dette vil medføre $\log_2(1/\varepsilon)$ iterasjoner av optimize_λ_1, som igjen vil kreve $\log_2(1/\varepsilon)^2$ iterasjoner av optimize_λ_2. Hver iterasjon vil kjøre funksjonen optimize_s.

Funksjonen optimize_s løser den vektede ratesumoptimaliseringen uavhengig på hver tone/delbånd. Dette må gjøres med et exhaustive søk, som søker igjennom alle kombinasjoner på det aktuelle delbåndet. For å gjøre dette kreves $K(b_{max} + 1)^2$ evalueringer av L_k hvis det er diskret bitloading som benyttes. Dette medfører at kompleksiteten til algoritmen blir $O(K(b_{max} + 1)^2 \log_2(1/\varepsilon)^2)$.

Hvis denne sammenlignes med den kompleksiteten som oppstår med Exhaustive search som ble diskutert tidligere i 1.4.1 må denne beregnes for $(b_{max} + 1)^{2K}$ bitloadingkombinasjoner. For et eksempel med ε på 10^{-10}, som skal være nok til få en tilfredsstillende nøyaktighet på resultatene vil $\log_2(1/\varepsilon)^2$ bli cirka 1100. Det er tydelig å se at for økende mulig bitloading på modemene (b_{max}) eller for økende antall delbånd (toner) vil Dual decomposition oppføre seg mye bedre.

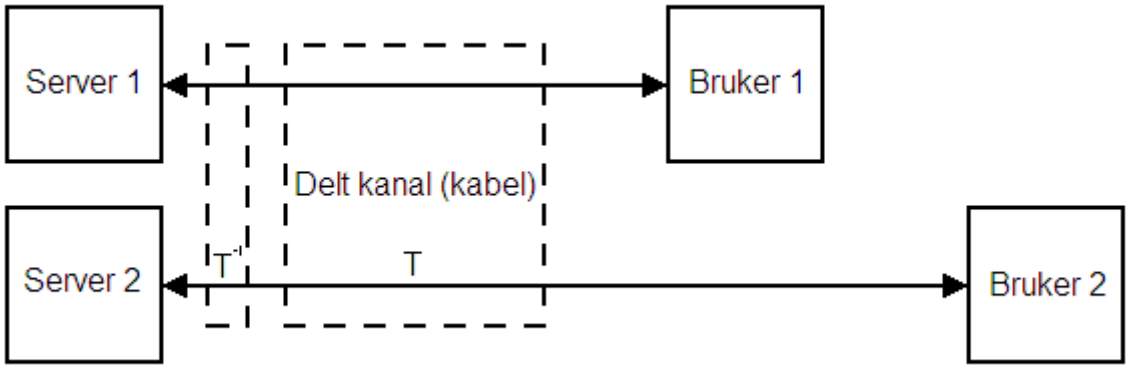
Det er verdt å merke seg at denne rapporten kun tar for seg to brukere. Med mindre modifikasjoner er en eventuell utvidning til flere enn to brukere mulig. Kompleksiteten er grundigere forklart og med flere (N) brukere i [6] kapittel IV D.

Ulempen med Dual decomposition er at kompleksiteten fremdeles er eksponentielt avhengig av antall brukere N og blir dermed vanskelig å beregne for stor N. Det er derfor ønskelig med andre fremgangsmåter. En nylig foreslått metode [7] er å reformulere det ikke-konvekse optimeringsproblemet i OSM til et ekvivalent globalt konkav minimeringsproblem.

1.5 DSM: Nivå 3 – Krysstalekansellering

Nivå 3 DSM også kalt ”vectoring” muliggjør de høyeste dataratene for DSL (i alle fall av det som er kjent i dag). Med ”vectoring” synkroniserer en vanlig DSLAM nedstrøm DSL overføring til en kjent DMT symbolklokke (enten 4,3125 kHz klokken til ADSL og VDSL eller dobbelbreddeklokken 8,625 kHz muligheten i VDSL). Dette vil igjen medføre den samme symbolklokken i oppstrøm. Det kalles vektor fordi hele kanalen blir gjenkjent med alle signalveier, og deretter blir signalene invertert med denne kanalvektoren. Dette vil fjerne all krysstale, og i noen tilfeller gjøre at man kan oppnå diversitetsgevinster på grunn av krysstalen.

”Vectoring” kan benyttes både enveis og toveis. Toveis vil oppføre seg på samme måte som MIMO (Multiple Input Multiple Output) for trådløse kanaler. En skisse som viser hvordan ”vectoring” blir i et forenklet blokkdiagram for to brukere er gitt i Figur 1.5.1. Her ser man kanalen, gitt av kanalvektoren T, og den inverse kanalen, gitt av vektoren T^{-1}.



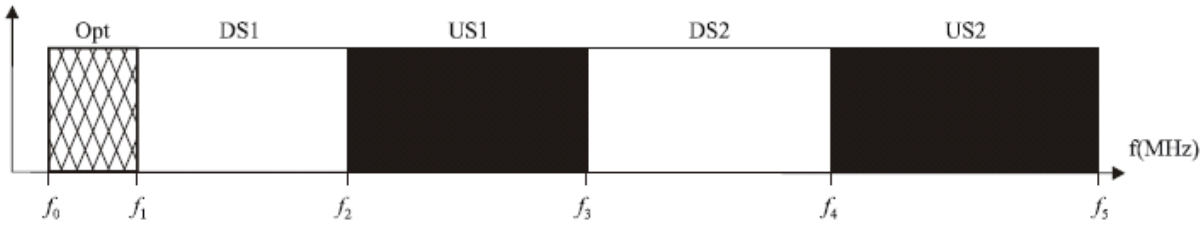
Figur 1.5.1 Krysstalekansellering

For mer informasjon angående nivå 3 anbefales det å lese annen litteratur, da denne rapporten ikke skal behandle dette.

1.6 VDSL (Very high speed Digital Subscriber Line)

G.993.1 VDSL [8] er en DSL teknologi som kan tilby raskere datatrafikk, både symmetrisk og asymmetrisk, over tvunnet kopperkabel. Dette gjør den mer attraktiv enn tidligere DSL modeller hvis man er ute etter høyere datarate for overføring av for eksempel HDTV.
VDSL:

- benytter et høyere frekvensbånd enn for eksempel ADSL
- benytter FDD (Frequency Division Duplexing) for å separere oppstrøm- og nedstrøm datatrafikk.
- benytter en firebånds plan som (se Figur 1.6.1) starter på 138 kHz og går opp til 12 MHz, der det er to nedstrømsbånd (DS1 og DS2) og to oppstrømsbånd (US1 og US2).



Figur 1.6.1 VDSL Frekvensbånd [8]

Frekvensene f_0, f_1, f_2, f_2, f_4 og f_5 bestemmes ut ifra hvilken båndplan man benytter. En båndplan kalles Bandplan A, tidligere kalt Plan 998. Denne planen er gjengitt i Figur 1.6.2.

	[MHz]	Antall toner/delbånd	Retning
$f_0 – f_1$	0,025-0,138	26	Bruk og retning valgfritt
$f_1 – f_2$	0,138-3,75	837	Nedstrøm
$f_2 – f_3$	3,75-5,2	336	Oppstrøm
$f_3 – f_4$	5,2-8,5	765	Nedstrøm
$f_4 – f_5$	8,5-12	811	Oppstrøm

Figur 1.6.2 VDSL Bandplan A / Plan 998 [8]

VDSL med DMT (Discrete Multi-Tone) modulasjon er blitt mer og mer brukt fordi den oppnår VDSLs operasjonelle krav mer effektivt enn QAM eller CAP. Symbolraten til DMT i VDSL og ADSL, f_s, settes lik frekvensbåndinndelingen, Δ_f, mellom hjelpebærebølgene (tonene). Denne er i de aller fleste tilfeller lik 4,3125 kHz, hvor de skal være sentrert ved frekvensene $f = k\,\Delta_f$. Toneindeksen k kan ha verdiene $k = 0, 1, 2, \ldots, N_{delbånd} - 1$. PSD er for de fleste tilfeller gitt til maksimum -52 dBm/Hz.

2 Simuleringer / Resultater

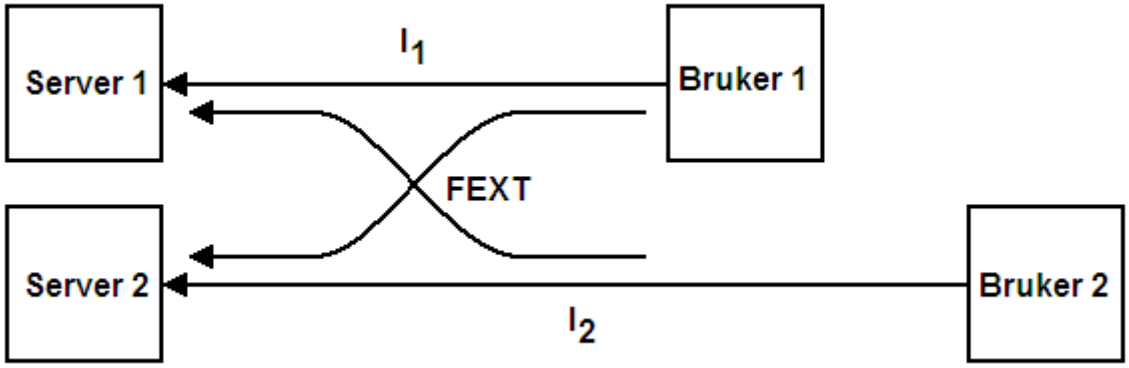
Målet med denne oppgaven er å evaluere om det noen forbedringer i beregningstiden og endringer i kompleksiteten med å beregne DSM med forskjellige algoritmer ved hjelp av simuleringer. Der det er beregnet for sammenligning er det testet med samme oppsett, med samme lengder, frekvenser og effekter.

En ting det er greit å huske på videre er at alle tider er tatt ved hjelp av MATLAB og en vanlig bærbar datamaskin (CPU: Intel Celeron M420@1,60GHz, RAM: 1GB). Dette medfører at tidene må sees på som relative fordi blant annet programmeringen og datamaskinen vil påvirke tidene noe. Skal noe lignende testes og/eller implementeres i et DSL-system vil dedikerte systemer være ønskelige.

For plottene der ikke alle delbåndene i oppstrømsbåndet er benyttet er resultatene skalert opp. Det er selvfølgelig ikke mulig å oppnå noen av de ratene som simuleringene gjør med bare noen få delbånd. Dette har blitt gjort for å få mer oversiktelige resultater som lettere kan sammenlignes.

2.1 System som skal simuleres

Det er valgt å simulere to brukere i VDSL. Grunnen til at det ble to brukere, og ikke flere er at det holder til å vise eventuelle forbedringer og det vil være ganske mye mindre komplekst enn flere. Overføringsmetode er som nevnt tidligere oppstrøm, der overføringen skjer i laveste oppstrømsbånd. En skisse av systemet er vist i Figur 2.1.1. Siden det kun er oppstrøm, vil det kun være FEXT krysstalenn som må taes hensyn til. Lengdene l_n er avstanden fra bruker n til server n. Det velges l_1 lik 0,5 km og l_2 lik 0,8 km noe som medfører at det kun vil oppstå krysstale de siste 0,5 km nærmest serverne, se figur.



Figur 2.1.1 Skisse av simulert system

Systemet skal simuleres med to forskjellige algoritmer (Exhaustive search og Dual decomposition) for DSM nivå 2 med varierende kompleksitet for å se hvordan dette påvirker kapasitetsregionen samt simulerings-/beregningstiden. Det velges relativt enkle oppsett for å få innsikt i hvordan de forskjellige algoritmene virker.

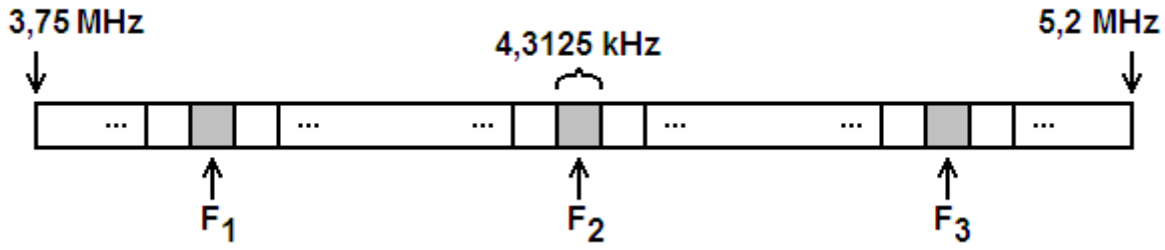
Først skal Exhaustive search prøves, der det søkes igjennom alle mulige kombinasjoner av forskjellige mulige PSD på modemene. Mulige PSD avhenger av bitloadingen på modemene. Ut i fra disse kombinasjonene skal rateregionen finnes.

Deretter skal Dual decomposition simuleres. Her er det en litt mer avansert algoritme som skal benyttes til å finne rateregionen. Denne bruker vektet optimalisering av ratesummen og finner ratene ved hjelp av å gå igjennom delbåndene.

Fra det som er diskutert i kapittel 1 Teori vil man anta at beregningstiden til Dual decomposition bør bli vesentlig mindre enn beregningstiden til Exhaustive search, i alle fall med stor nok bitloading og/eller mange nok benyttede delbånd.

FEXT modellen som skal brukes er "Worst case FEXT", det vil si $10^{-fext1dB/10} F^2 L_{overlapp}$ der fext1dB er lik 45 dB ved 1 MHz og 1 km. F er frekvensen i MHz og $L_{overlapp}$ er total lengde i km med FEXT. Total margin skal være 3 dB (for Shannon) + 1 dB (sikkerhetsmargin) = 4 dB. Støyeffekten for hvit støy settes lik -140dBm/Hz som er vanlig å anta i slike simuleringer. Signalet dempes med $\sqrt{F}$ og er gitt som 22,5 dB/km ved 1 MHz. Effektdempingen blir dermed proporsjonal med $e^{-2\alpha L}$, der $\alpha = 22{,}5\frac{\log 10}{20}\sqrt{F}$ og L er lengden på det tvunnet paret.

Siden det er VDSL som skal brukes vil det for nivå 2 bety at DMT symbolrate settes lik $f_s = \Delta_f = 4{,}3125$ kHz. For enklere implementering skal overføringen for Exhaustive search først skje på tre av de mulige delbåndene i oppstrømsbåndet, dvs ved bruk av tre hjelpebærebølger. Symmetrisk inndeling vil gi en inndeling skissert i Figur 2.1.2 med tre bærebølger for delbåndene, F_1, F_2 og F_3. Eventuelle flere delbånd vil følge naturlig ut fra symmetri.



Figur 2.1.2 Tre delbånd fra VDSL

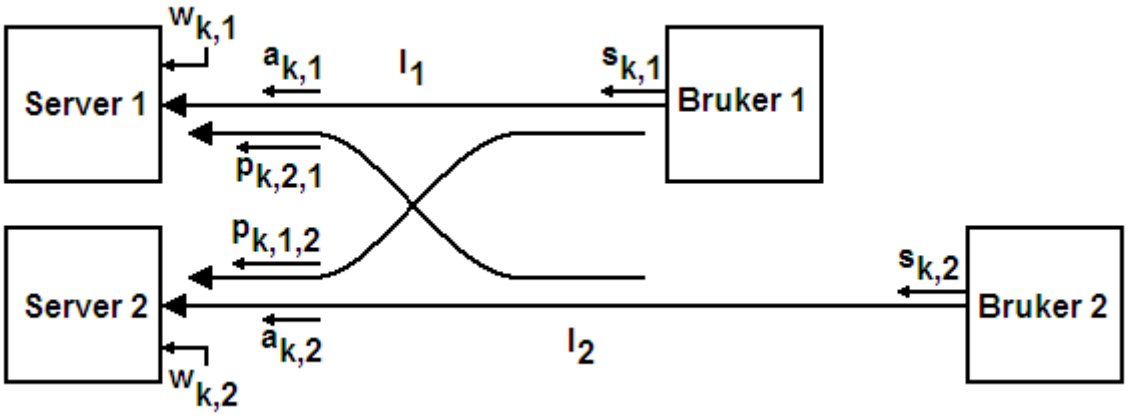
De tre bærebølgene regnes enkelt ut ved å dele opp båndet i alle de $\frac{5{,}2MHz - 3{,}75MHz}{4{,}3125MHz} \approx 336$ delbåndene, noe som medfører at vi får

$$F_1 = 3{,}75MHz + \left(\frac{336}{6}1\right)4{,}3125kHz = 3{,}9915MHz$$

$$F_2 = 3{,}75MHz + \left(\frac{336}{6}3\right)4{,}3125kHz = 4{,}4745MHz$$

$$F_3 = 3{,}75mHz + \left(\frac{336}{6}5\right)4{,}3125kHz = 4{,}9575MHz$$

Figur 2.1.3 viser hvordan det totale system blir innsatt variablene fra Formel 1.4.1 der bitraten regnes ut.

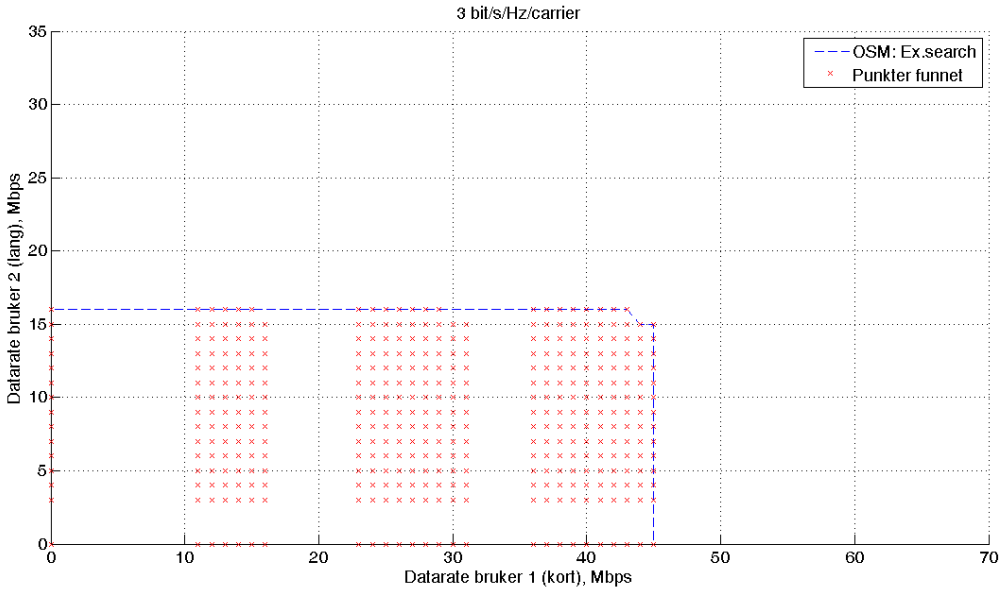


Figur 2.1.3 Forklarende blokkdiagram for rateregning

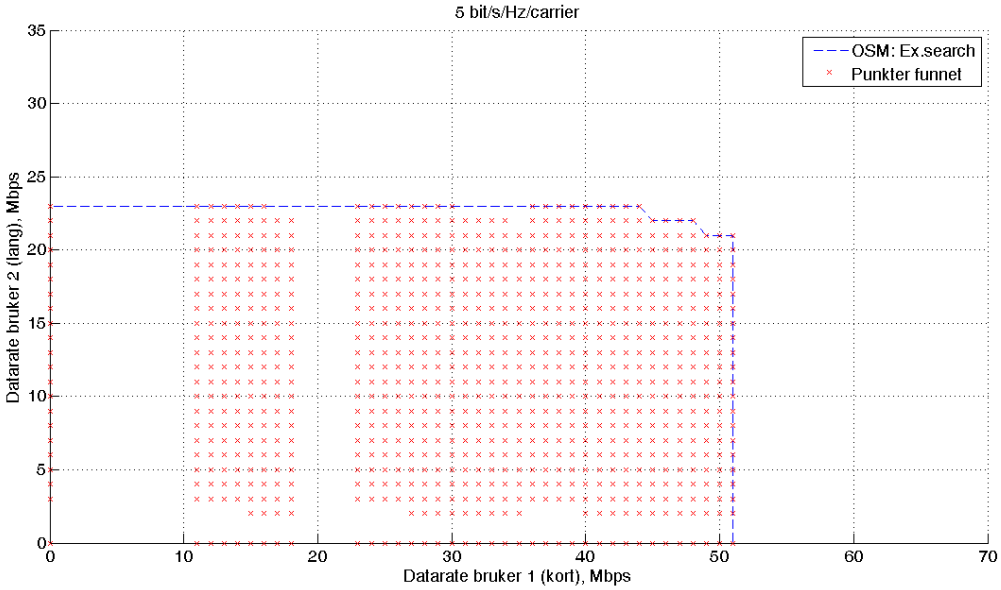
2.2 Exhaustive search

2.2.1 Tre delbånd

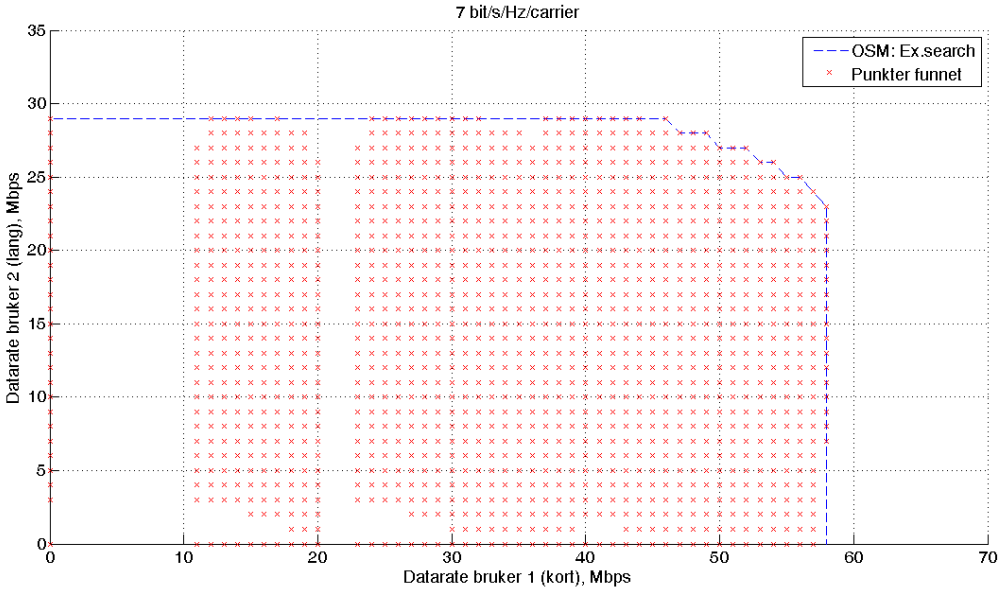
Systemet ble initialisert og simulert med Exhaustive search algortimen. For å se hvordan den optimale kapasitetsregionen ble for forskjellige bitloadinger på modemene, ble det testet med bitloading på 3, 5 og 7 bit/s/Hz/carrier. Noe som da førte til 16, 36 og 64 aktuelle effekttettheter per delbånd. For å få vist hvor fort Exhaustive search øker i kompleksitet og for å ha noe mer å sammenligne med Dual decomposition, blir MATLAB brukt til å finne ut hvor lang tid hver beregning/simulering tar ved hjelp av clock-funksjonen.



Figur 2.2.1 Exhaustive search med 3 bit/s/Hz/carrier



Figur 2.2.2 Exhaustive search med 5 bit/s/Hz/carrier



Figur 2.2.3 Exhaustive search med 7 bit/s/Hz/carrier

Bitloading	Tid
3 bit/s/Hz/carrier	0,6880 s
5 bit/s/Hz/carrier	36,9530 s
7 bit/s/Hz/carrier	1475,4220 s (24m 35,4220 s)

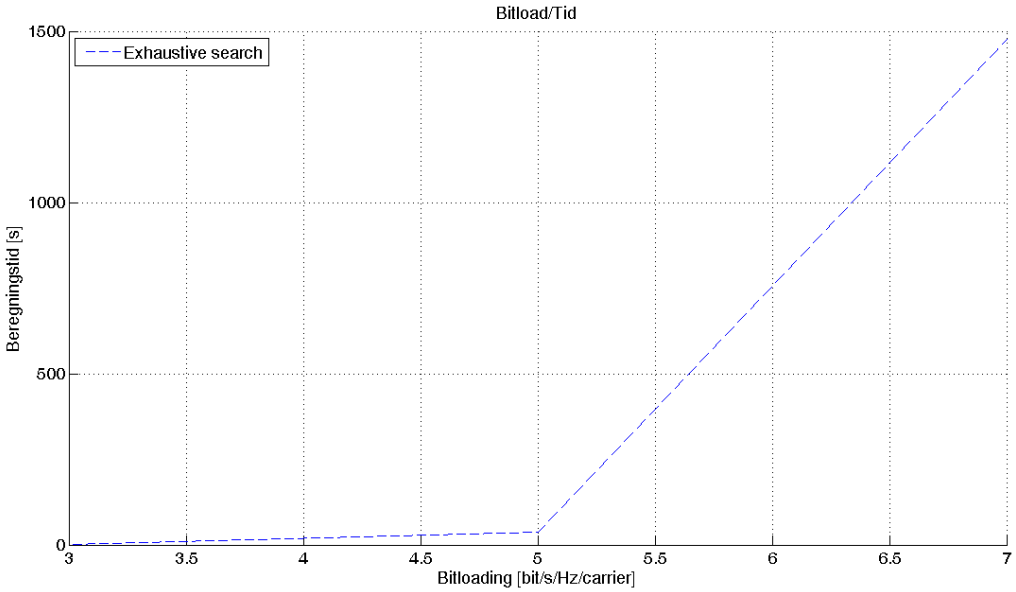
Figur 2.2.4 Simuleringstider Exhaustive search

Hvis plottene av mulige ratekombinasjoner i Figur 2.2.1, Figur 2.2.2 og Figur 2.2.3 undersøkes vises resultatet av at dette er en diskret optimalisering, med bare et visst antall valgbare ratekombinasjoner beregnet med Exhaustive search. Grunnen til at den blir hakkete er dermed på grunn av de diskrete verdiene samt det faktum at det kun testes for tre av 336

mulige bånd. Hvilken ratekombinasjon som velges avhenger av hvor store rater de to brukerne vil oppnå.

Man ser at det er bruker 1 som oppnår de høyeste datratene. Dette skjer da man kan observere at near-far effekt i dette tilfellet vil være av betydning. Near-far effekt betyr at krysstalen som bruker 1 ”påfører” bruker 2 er mindre i effekt enn krysstalen bruker 2 ”påfører” bruker 1. Dette på grunn av at bruker 1 har kortere lengde på kabelen enn bruker 2. Hadde brukerne hatt like avstander fra serveren ville rateregionen ha vært symmetrisk.

Det observeres at ytelsen øker med økende bitloading. Dette er en ganske naturlig fremgang, i alle fall opp til det nivået der maks total effekt blir oppnådd. Man kan også se at beregningstidene for de forskjellige bitloadingene vist i Figur 2.2.4 øker betraktelig for økende bitloading. Dette stemmer med teorien som sier at kompleksiteten er avhengig av $O((b_{max} + 1)^{2K})$. For henholdsvis 3, 5 og 7 i bitloading blir denne O(4096), O(46656) og O(262144). Et plot av simuleringstid med hensyn på bitloading er gitt i Figur 2.2.5.



Figur 2.2.5 Bitload/Tid Exhaustive search

OSM Exhaustive search vil som diskutert i teorien være det beste alternativet uten krysstalekansellering og vil derfor være en høyst ønskelig teknologi å implementere. Måten datarateverdiene er beregnet i dette prosjektet er ikke gjennomførbart for et praktisk system, da kompleksiteten vil bli stor med høy bitloading, mange delbånd og mange brukere. Det er derfor ønskelig med mer effektive algoritmer, som for eksempel Dual decomposition som er diskutert i teorien.

Det at OSM er optimal betyr ikke at det er det beste å implementere. For praktiske tilfeller kan tilnærminger være bedre for å unngå kompleksitet. OSM kan i disse tilfellene heller benyttes som et øvre mål for ratene.

2.2.2 Fire delbånd

Noe som kan være nyttig å se på er hvordan beregningstiden for Exhaustive search reagerer med økende antall delbånd. Derfor økes antall delbånd fra tre til fire med uendret bitloading på 3 bit/s/Hz/carrier. Dette er gjengitt i Figur 2.2.6.

Bitloading	Tid
3 bit/s/Hz/carrier	81,9840 s (1m 21,9840 s)

Figur 2.2.6 Simuleringstider Exhaustive search med fire delbånd

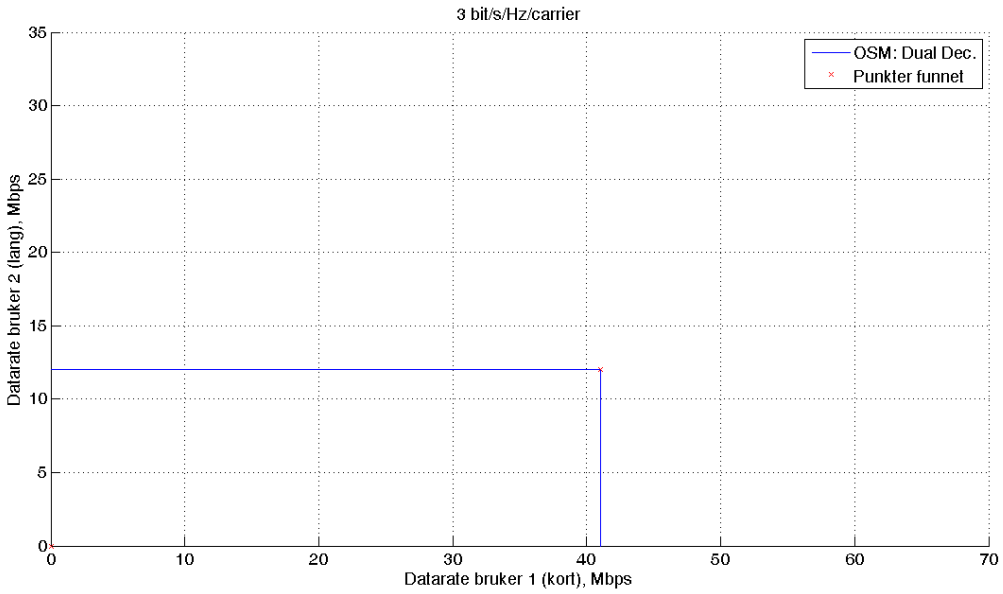
Det kan observeres med denne tiden at beregningstiden øker mindre når bitloadingen økes med 2 bit/s/Hz/carrier fra 3 bit/s/Hz/carrier til 5 bit/s/Hz/carrier uten å endre antall delbånd enn når antall delbånd økes med ett fra tre til fire med samme bitloading på 3 bit/s/Hz/carrier. Dette stemmer med det som er diskutert i teorien, der kompleksiteten er gitt eksponesielt avhengig av antall delbånd som benyttes, det vil si $O((b_{max} + 1)^{2K}$ for to brukere med Exhaustive search. Settes det inn verdier for de to tilfellene vil kompleksiteten øke fra $O((3 + 1)^{2*3}) = O(4096)$ til $O((5 + 1)^{2*3}) = O(46656)$ for økningen i bitloading og $O((3 + 1)^{2*4}) = O(65536)$ for økningen i antall delbånd. Simuleringer med flere delbånd vil dermed ta ekstremt lang tid.

2.3 Dual decomposition

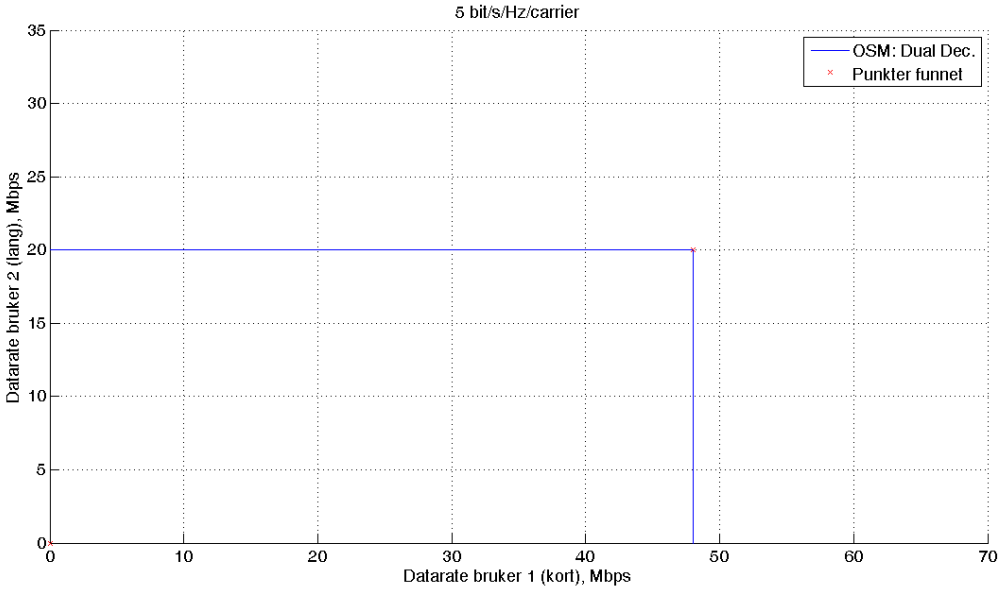
For å understreke hvor mye bedre det er å benytte Dual decomposition i stedet for Exhaustive blir først Dual decomposition simulert med tre delbånd for sammenligningens skyld. Etterpå simuleres Dual decomposition med flere, og til slutt alle, delbåndene i oppstrømsbåndet benyttet. Dette for å vise at Dual decomposition er en metode som kan benyttes uten stor datakraft.

2.3.1 Tre delbånd

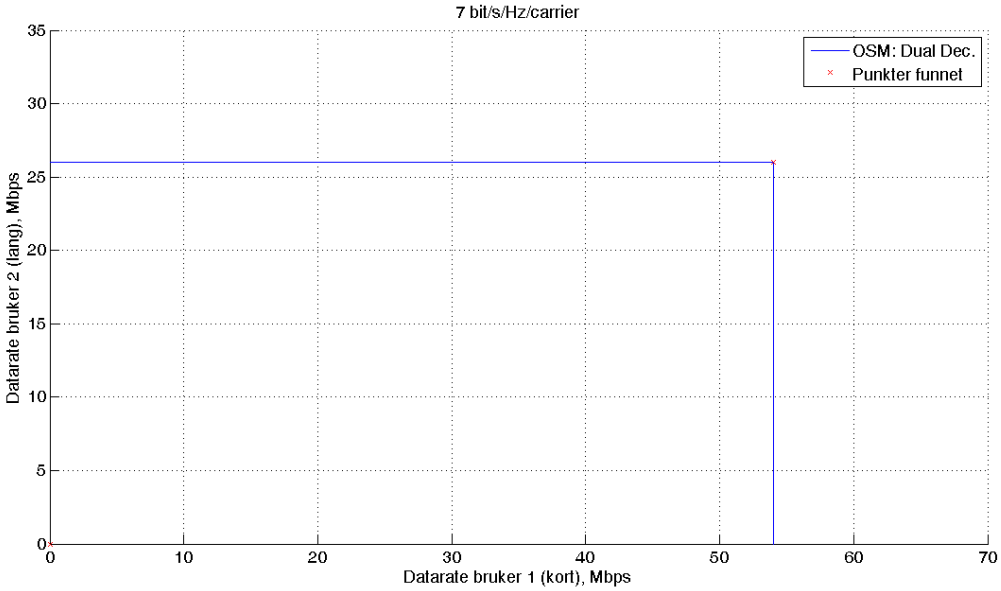
Først ble systemet med tre delbånd implementert for simulering. Dette gjøres, som sagt, for å ha noe som kan sammenlignes med Exhaustive search. I første omgang er det bitloading på 3, 5 og 7 bit/s/Hz/carrier som benyttes, der oppløsningen på plottene er (70,35). Så blir det simulert for 10, 12 og 15 bit/s/Hz/carrier der oppløsningen er (100,100).



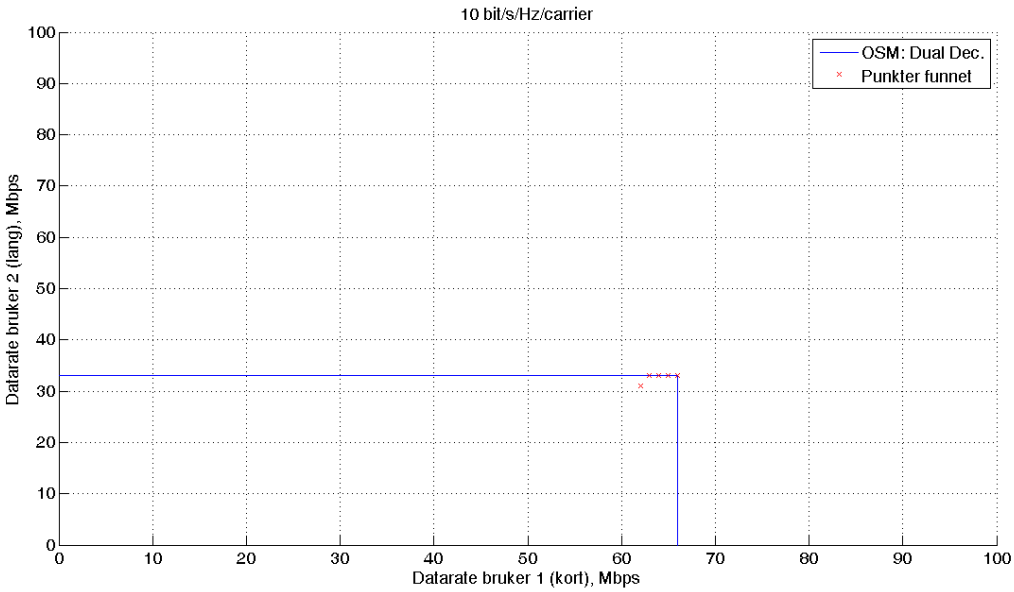
Figur 2.3.1 Dual decomposition med 3 bit/s/Hz/carrier



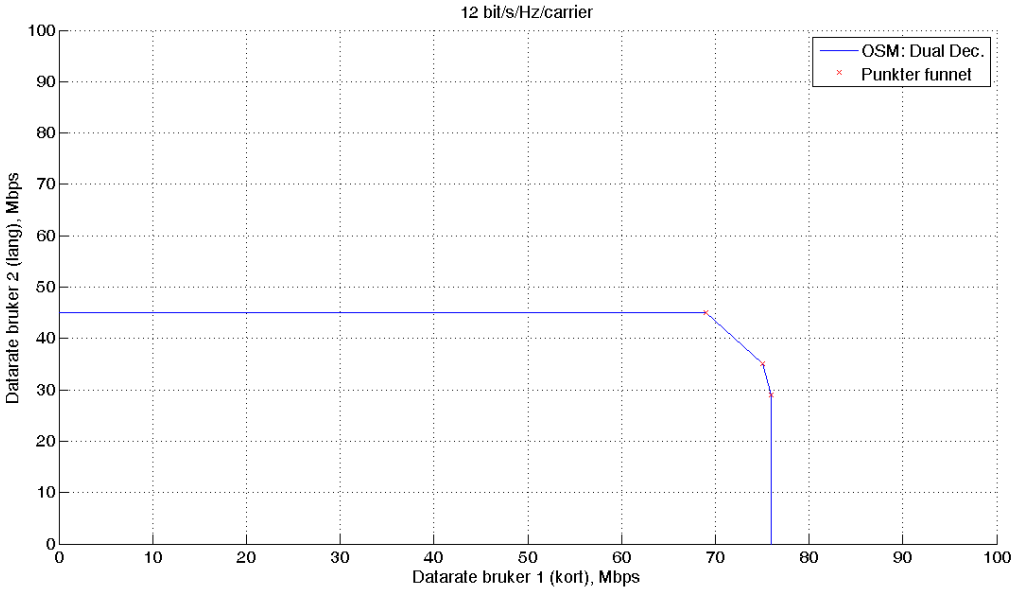
Figur 2.3.2 Dual decomposition med 5 bit/s/Hz/carrier



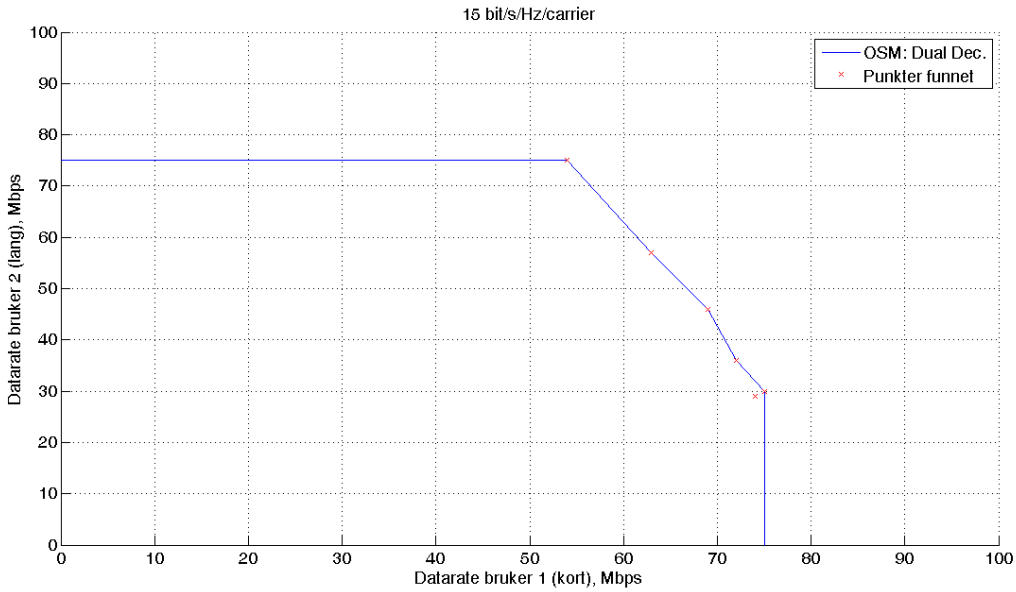
Figur 2.3.3 Dual decomposition med 7 bit/s/Hz/carrier



Figur 2.3.4 Dual decomposition med 10 bit/s/Hz/carrier



Figur 2.3.5 Dual decomposition med 12 bit/s/Hz/carrier



Figur 2.3.6 Dual decomposition med 15 bit/s/Hz/carrier

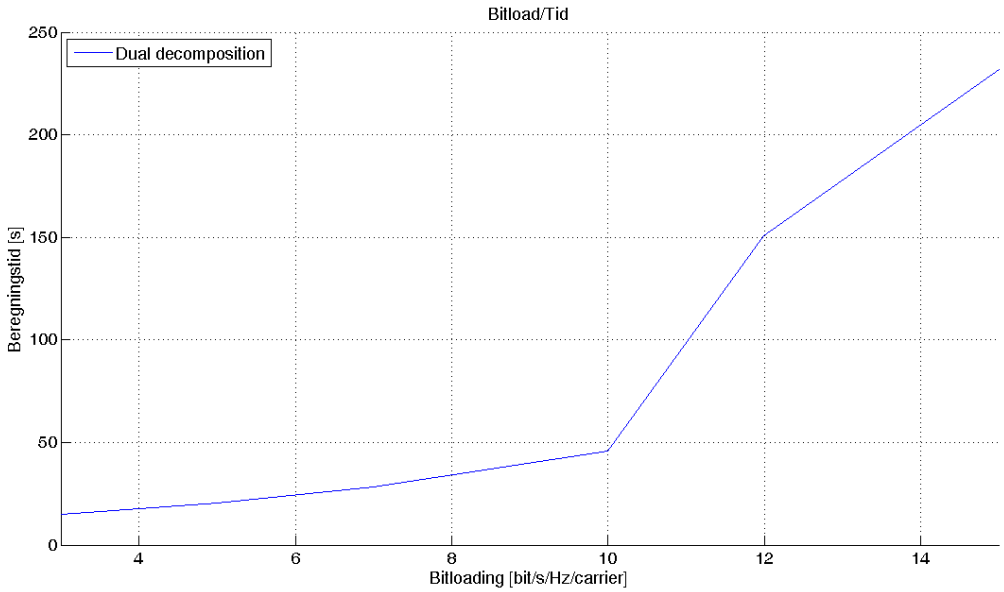
Bitloading	Tid
3 bit/s/Hz/carrier	14,7180 s
5 bit/s/Hz/carrier	20,3280 s
7 bit/s/Hz/carrier	28,4530 s
10 bit/s/Hz/carrier	45,7180 s
12 bit/s/Hz/carrier	150,8120 s (2m 30,8120 s)
15 bit/s/Hz/carrier	231,9220 s (3m 51,9220 s)

Figur 2.3.7 Simuleringstider Dual decomposition

Som det kan observeres ut i fra disse plottene (Figur 2.3.1 til Figur 2.3.6), ser man at Dual decomposition finner mange færre punkter enn Exhaustive search. Dette er fornuftig med tanke på at kompleksiteten er såpass mye mindre. Det kan også observeres at noen av punktene sammenfaller, noe som er fornuftig med tanke på begrensningene i total utsendt effekt.

En annen observasjon er at når bitloadingen blir større blir ytelsen større, akkurat som Exhaustive search. I disse plottene vises det tydelig at total effektbegrensning nås ved stor bitloading ved at rateregionen blir ”kuttet av” (gjelder spesielt for 12 og 15 bit/s/Hz/carrier, dvs. Figur 2.3.5 og Figur 2.3.6).

Hvis man ser på beregningstidene gitt bitload i Figur 2.3.7 ser man at tiden stiger for økende bitloading, men ikke så veldig dramatisk med tanke på beregningsmengden. Disse tidene er vist som en graf i Figur 2.3.8.



Figur 2.3.8 Bitload/Tid Dual decomposition

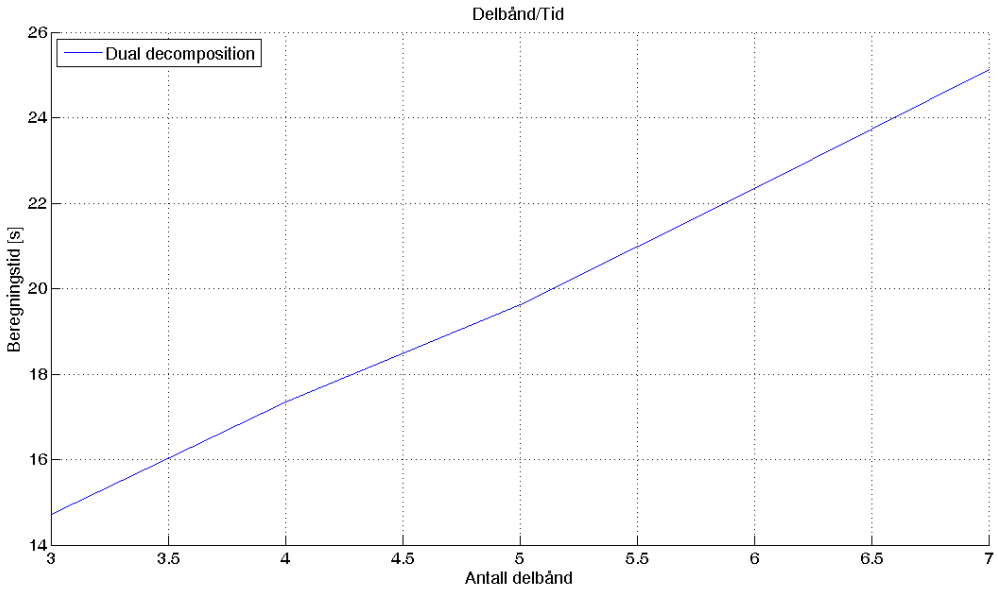
2.3.2 Flere delbånd

For å se hvordan Dual decomposition sin beregningstid utvikler seg med flere delbånd, taes fire simuleringer av Dual decomposition med. Fire, fem, seks og syv delbånd blir tatt i bruk med bitloading på 3 bit/s/Hz/carrier. Disse tidene er oppført i Figur 2.3.9.

Bitloading	Tid	Antall delbånd
3 bit/s/Hz/carrier	17,3590 s	4
3 bit/s/Hz/carrier	19,6250 s	5
3 bit/s/Hz/carrier	22,3430 s	6
3 bit/s/Hz/carrier	25,1250 s	7

Figur 2.3.9 Simuleringstider Dual decomposition med fire, fem, seks og syv delbånd

Et plot av beregningstid gitt delbånd (med bitloading på 3 bit/s/Hz/carrier) er gitt i Figur 2.3.10. Tiden det tar å simulere med bitloading på 3 bit/s/Hz/carrier og tre delbånd leses ut fra Figur 2.3.7.



Figur 2.3.10 Delbånd/Tid Dual decomposition

I dette enkle plottet kan man se at Dual decomposition oppfører seg nesten lineært. Dette stemmer overens med det som ble lagt fram i teorien. Kompleksiteten (og dermed beregningstiden) var gitt som proporsjonal med antall delbånd K, det vil si $O(K(b_{max} + 1)^2 \log_2(1/\varepsilon)^2)$ for to brukere.

2.3.3 Alle delbånd

For å vise hva Dual decomposition er god for taes simulering av rateregionene med 10, 12 og 15 der alle delbåndene i hele oppstrømsbåndet er tatt med, det vil si frekvensene

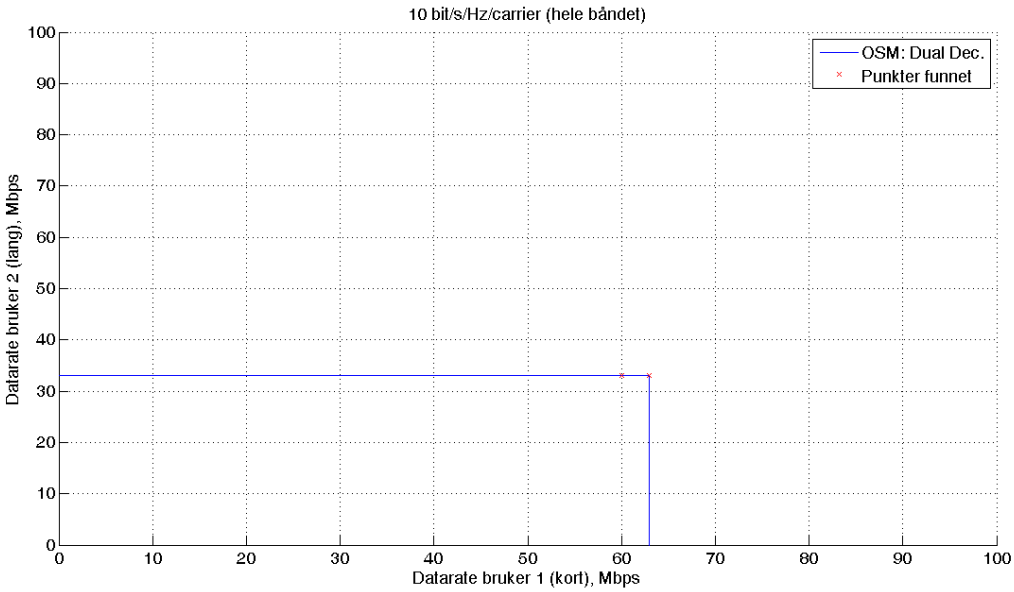
$$F_1 = 3{,}75MHz + \left(\frac{1}{2} + 0\right)4{,}3125kHz \approx 3{,}7522MHz$$

$$F_2 = 3{,}75MHz + \left(\frac{1}{2} + 1\right)4{,}3125kHz \approx 3{,}7565MHz$$

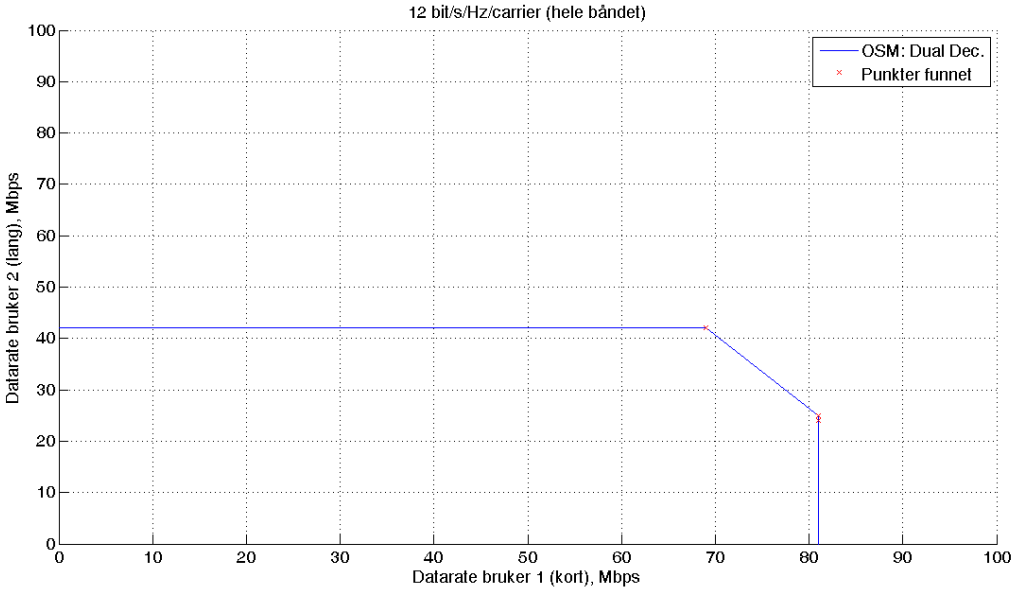
$$\vdots$$

$$F_{336} = 3{,}75MHz + \left(\frac{1}{2} + 335\right)4{,}3125kHz \approx 5{,}1968MHz$$

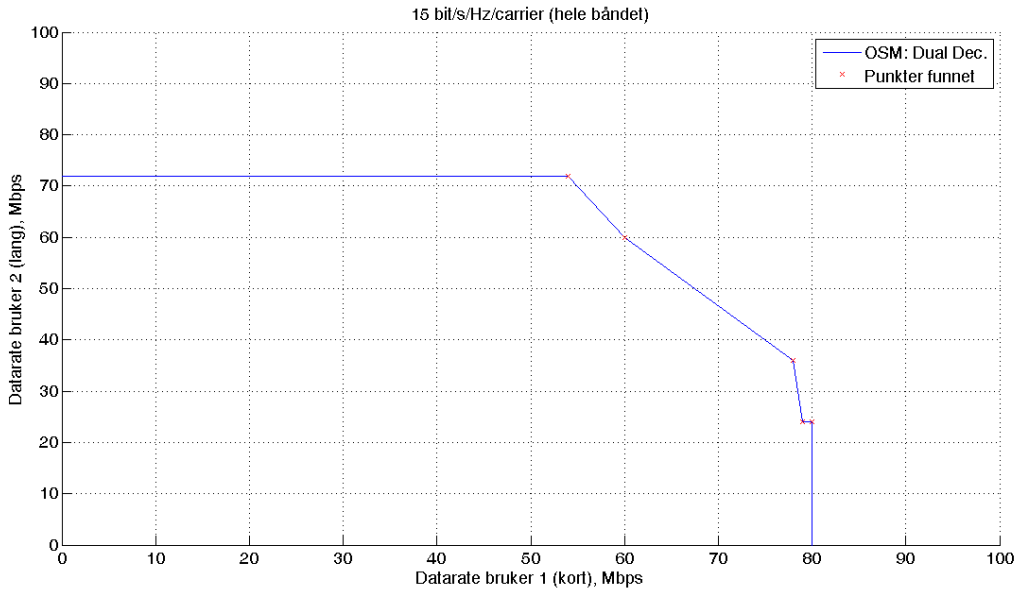
Dette vil si at det blir brukt $\frac{336}{3} = 112$ ganger flere frekvenser i utregningen, noe som ville ha vært så godt som ugjennomførbart med Exhaustive search. På grunn av økningen i antall frekvenser justeres ε opp til 10^{-2} slik at nøyaktigheten senkes. Dette vil minske kompleksiteten slik at beregningene tar kortere tid, men resultatene vil da også bli mindre optimale.



Figur 2.3.11 Dual decomposition med 10 bit/s/Hz/carrier (hele oppstrømsbåndet)



Figur 2.3.12 Dual decomposition med 12 bit/s/Hz/carrier (hele oppstrømsbåndet)



Figur 2.3.13 Dual decomposition med 15 bit/s/Hz/carrier (hele oppstrømsbåndet)

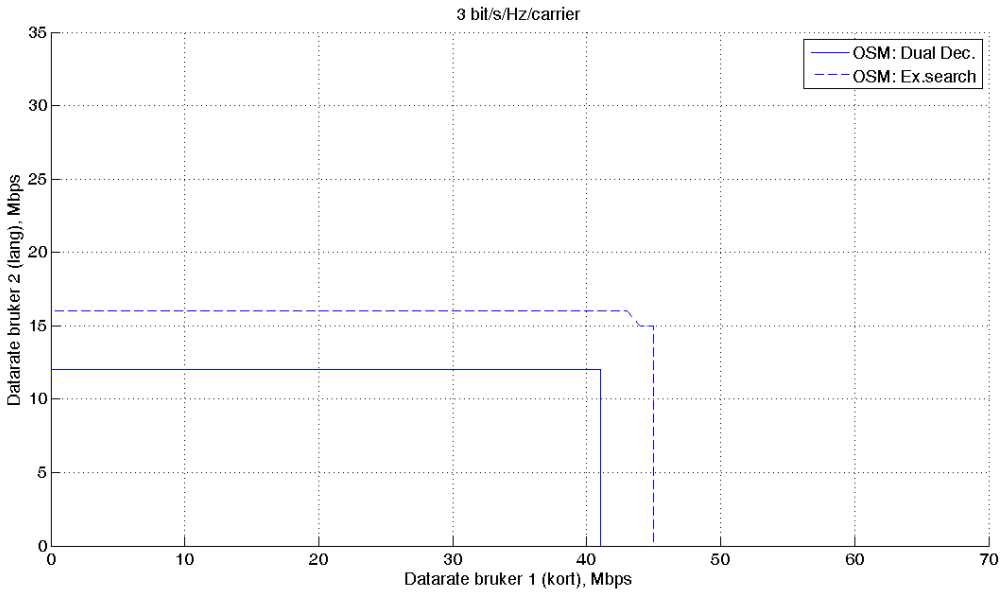
Bitloading	Tid
10 bit/s/Hz/carrier	217,0940 s (3m 37,0940 s)
12 bit/s/Hz/carrier	690,2970 s (11m 30,2970 s)
15 bit/s/Hz/carrier	1121,2030 s (18m 41,2030 s)

Figur 2.3.14 Simuleringstider Dual decomposition (hele oppstrømsbåndet)

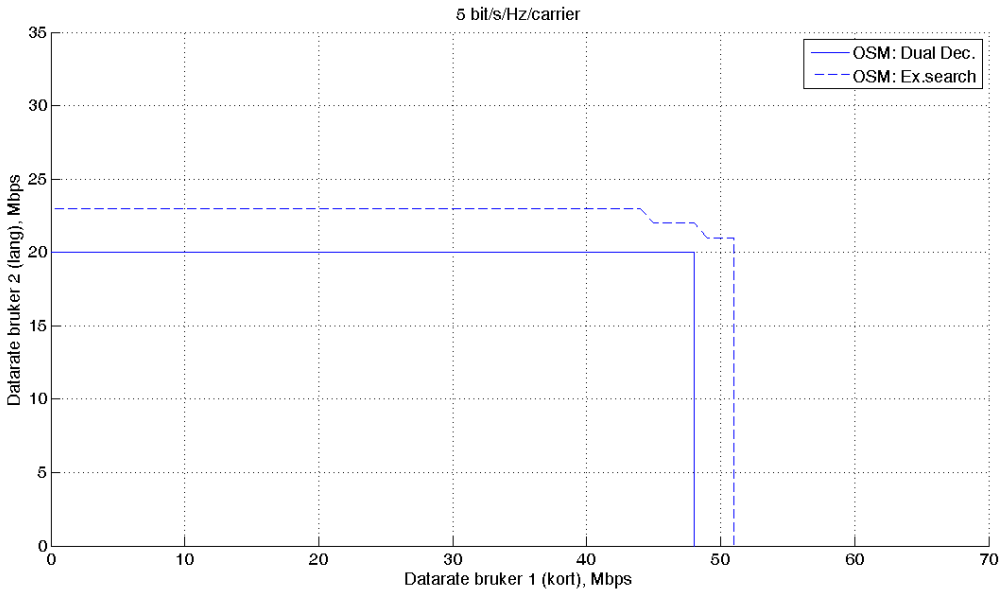
Som det observeres med Figur 2.3.11, Figur 2.3.12 og Figur 2.3.13 er at de ligner på tilsvarende simuleringer med færre delbånd. Den største forskjellen er at det ble funnet flere punkter i disse simuleringene. Det som kanskje er mest interessant med disse simuleringene er at tiden det tar å beregne med bitloading på 15 (se Figur 2.3.14) er mindre enn å beregne med bitloading på 7 med Exhaustive search når bare tre delbånd blir benyttet som det står i Figur 2.2.4.

2.4 Sammenligning Exhaustive search og Dual decomposition

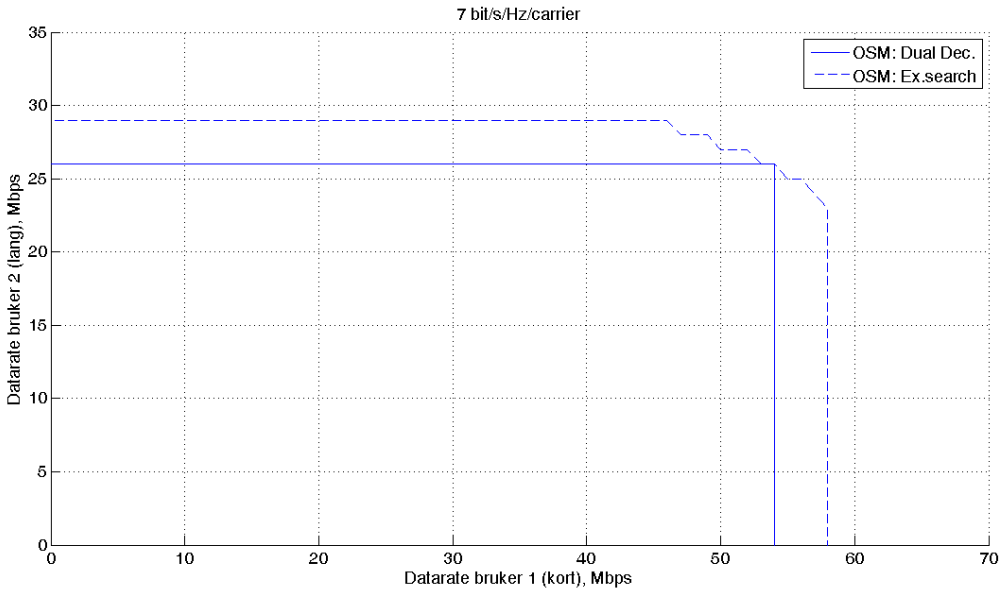
2.4.1 Tre delbånd



Figur 2.4.1 Exhaustive search og Dual decomposition med 3 bit/s/Hz/carrier

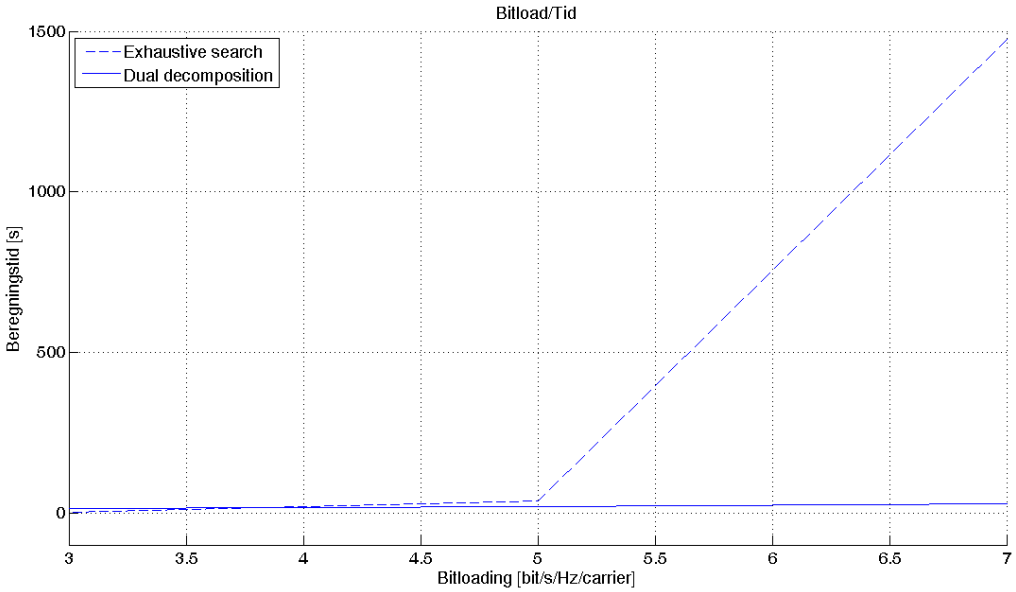


Figur 2.4.2 Exhaustive search og Dual decomposition med 5 bit/s/Hz/carrier



Figur 2.4.3 Exhaustive search og Dual decomposition med 7 bit/s/Hz/carrier

For både 3, 5 og 7 bit/s/Hz/carrier fra figur Figur 2.4.1, Figur 2.4.2 og Figur 2.4.3 kan det observeres at Exhaustive search vil finne et litt større område med bedre oppløsning enn Dual decomposition. Dette er et resultat som er fornuftig med tanke på at Exhaustive search gjør en grundigere jobb med søkingen. At Dual decomposition ikke finner absolutt alle arbeidspunkt er gjerne en liten pris å betale med tanke på at beregningstiden blir såpass mye mindre.



Figur 2.4.4 Bitload/Tid Exhaustive search og Dual decomposition

I Figur 2.4.4 er Figur 2.2.5 slått sammen med litt av Figur 2.3.8. Plottet viser hvordan økende bitloading forandrer beregningstiden for Exhaustive search og Dual decomposition for bitloading på 3, 5 og 7 bit/s/Hz/carrier. Det som kan legges merke til her er at for 3 bit/s/Hz/carrier er Exhaustive search forholdsvis mye raskere ($\frac{14{,}7180\ s}{0{,}6880\ s} \approx 21$ ganger) enn

Dual decomposition. Dette er fordi kompleksiteten til Exhaustive search ikke er så stor ennå da det kun benyttes to brukere og tre delbånd. Heller ikke på 5 bit/s/Hz/carrier er forskjellen mellom de to blitt veldig stor (Dual decomposition er $\frac{20{,}3280\ s}{36{,}9530\ s} \approx 1{,}8$ ganger raskere enn Exhaustive search). Det er når bitloadingen når 6-7 at fordelen med Dual decomposition kommer frem. Da blir beregningstiden for Exhaustive search hele $\frac{1475{,}4220\ s}{28{,}4530\ s} \approx 52$ ganger større. Dette vil bare fortsette for økende bitloading. Siden bitloadingen på dagens modem er rundt 10-15 er det ganske tydelig at dette vil bli upraktisk for Exhaustive search etter hvert.

En annen måte å se hvordan Dual decomposition utvikler seg i forhold til Exhaustive search er å se på det tilfellet der antall delbånd øker, noe som blir gjort i 2.4.2.

2.4.2 Fire delbånd

Siden det vil bli mye de samme resultatene for tre delbånd som for fire når rateberegningene utføres, ble fire delbånd tatt med for å vise hvor mye beregningstidene endrer seg når antall delbånd økes med ett.

For Exhaustive search er økningen enorm. Fra Figur 2.2.4 og Figur 2.2.6 observeres det at beregningstiden øker med en faktor på $\frac{81{,}9840\ s}{0{,}6880\ s} \approx 120$, mens den for Dual decomposition fra Figur 2.3.7 og Figur 2.3.9 beregnes at faktoren bare er på $\frac{17{,}3590\ s}{14{,}7180\ s} \approx 1{,}2$. Også her, som for økende bitloading, ser man at Exhaustive search vil være temmelig upraktisk. I alle fall med tanke på at det i hele oppstrømsbåndet befinner seg 336 delbånd.

2.5 MATLAB kode

All koding, simulering og plotting ble utført i MATLAB. Utgangspunktet for koden var DSLsim opprinnelig laget av professor Nils Holte ved NTNU våren 2001.

For begge delene av simuleringen ble det essensielle plukket ut av koden, da det ikke var bruk for alle de funksjonene og tilleggene i koden skrevet av Holte. Denne koden er gjengitt i vedleggene 6.1, 6.2, 6.3, 6.4, 6.5, 6.6, 6.7, 6.8 og 6.9. En liten beskrivelse av hver m-fil er også å finne der. Ved hjelp av m-filene kan alle plottene som er laget i denne rapporten gjengis på nytt.

3 Konklusjon

Som vist er oppnåelige datarater for xDSL (for eksempel VDSL) begrenset oppad av krysstalen mellom kopperkablene. Siden dynamisk frekvensbruk kan love store ytelsesforbedringer er DSM et omfattende forskningsområde som det jobbes mye med i disse dager. Høyst trolig vil forskjellige former av DSM bli implementert i flere og flere nettverk i årene som kommer.

For de simuleringene som er gjort i denne oppgaven er det benyttet sentraliserte kontrollere uten å innføre krysstalekansellering. Det var tidligere vist at dette kunne medføre veldig komplekse beregninger. Ved å sammenligne algoritmene Exhaustive search og Dual decomposition var det mulig å se et stort forbedringspotensial i beregningstidene for oppnåelige datarater. Grunnen til dette er at kompleksiteten til Dual decomposition er lineær med hensyn på antall benyttede delbånd, mens Exhaustive search er eksponentiell.

3.1 Videre arbeid

Videre arbeid innen dette området vil være å gjøre flere undersøkelser for Dual decomposition. Dette gjelder helst i det tilfellet at det er få brukere. Dersom flere brukere er ønsket må mindre optimale algoritmer undersøkes. Dette fordi kompleksiteten fremdeles er eksponentielt avhengig av antall brukere. Et eksempel på en algoritme som skal være mindre kompleks er Concave minimization [7].

Å bruke dynamisk frekvensdeling for å oppnå høyere datarater er kjent. Spørsmålet er; når vil dette bli såpass enkelt og billig at det vil ta over for statiske løsninger?

4 Figurliste

5 Referanser

[1] N. Holte, “Broadband Communication in existing copper cables by means of xdsl system – a tutorial” Trondheim, 2001
[2] S. Haykin, “Communication Systems”, 4th edition, John Wiley & Sons, Inc., 2001
[3] J. Barry, E. Lee, D og Messerschmitt, “Digital communication”, 3rd edition, Springer Science+Business Media, Inc., 2004
[4] J. M. Cioffi and M. Mohseni, “Dynamic Spectrum Mangement,” ISSLS 3/04, Edinburgh, Scotland.
[5] http://www.stanford.edu/group/cioffi/dsm/, våren 2008
[6] R. Cendrillon, M. Moonen, J. Verliden, T. Bostoen og W. Yu, ”Optimal Multi-user spectrum balancing for digital subscriber lines”, Communications, IEEE Transactions on, Volume 54, Issue 5, May 2006, Pages:922 - 933
[7] Y. Xu, S. Panigrahi og T. Le-Ngoc, “A concave minimization approach to dynamic spectrum management for digital subscriber lines”, Communications, 2006 IEEE International Conference on, Volume 1, June 2006, Pages:84 - 89
[8] ITU-T “G.993.1 Very high speed digital subscriber line transceivers”, International Telecommunication Union, 06/2004
[9] http://en.wikipedia.org/wiki/Bisection_method, våren 2008

6 Vedlegg

6.1 run_plot.m

run_plot.m er hovedfilen som kjører alle valgte simuleringer, og plotter de. Denne tar også tiden på beregningene.

```
clear all;
global F antallbit epsilon;

%% Initialisering
run = 2;            % EX:1, DD:2, EX & DD:3
plotting = 0;       % Nei:0, Ja:1
B = 4.3125*10^-3;   % Båndbredde delbånd
frekvensvalg = 7;   % Tre f:3, Alle f:0, Annen f:valgfri
epsilon = 10^-10;   % Nøyaktighet Lagrange
antallbit = 3;      % Bitloading

%% Forberedelser
if frekvensvalg == 3
    F = [3.9915 4.4745 4.9575];
elseif frekvensvalg == 0
    F = (3.75+B/2 : B : 5.2);
else
    F = (3.75+336/frekvensvalg/2*B : B*336/frekvensvalg : 5.2);
end
if length(F) < 336
    if antallbit == 3
        tittel = '3 bit/s/Hz/carrier';
        akseverdier = [0 70 0 35];
    elseif antallbit == 5
        tittel = '5 bit/s/Hz/carrier';
        akseverdier = [0 70 0 35];
    elseif antallbit == 7
        tittel = '7 bit/s/Hz/carrier';
        akseverdier = [0 70 0 35];
    elseif antallbit == 10
        tittel = '10 bit/s/Hz/carrier';
        akseverdier = [0 100 0 100];
    elseif antallbit == 12
        tittel = '12 bit/s/Hz/carrier';
        akseverdier = [0 100 0 100];
    elseif antallbit == 15
        tittel = '15 bit/s/Hz/carrier';
        akseverdier = [0 100 0 100];
    end
else
    if antallbit == 10
        tittel = '10 bit/s/Hz/carrier (hele båndet)';
        akseverdier = [0 100 0 100];
    elseif antallbit == 12
        tittel = '12 bit/s/Hz/carrier (hele båndet)';
        akseverdier = [0 100 0 100];
    elseif antallbit == 15
        tittel = '15 bit/s/Hz/carrier (hele båndet)';
        akseverdier = [0 100 0 100];
    end
end

%% Simulering og plot
start_tid = clock;
if run == 1
    % Exhaustive Search
    exhaustive;
    stopp_tid = clock;
```

```
    if plotting == 1
        set(gcf,'DefaultLineLineWidth',1);
        set(gcf,'DefaultAxesFontSize',14);
        hold on;
        grid on;
        xlabel('Datarate bruker 1 (kort), Mbps');
        ylabel('Datarate bruker 2 (lang), Mbps');
        plot(rate1verdier,rate2verdier,'--');
        plot(ratebruker1,ratebruker2,'rx');
        legend('OSM: Ex.search','Punkter funnet','location','northeast')
        axis(akseverdier);
        title(tittel);
    end
elseif run == 2
    % Dual Decomposition
    dualdec;
    stopp_tid = clock;
    if plotting == 1
        set(gcf,'DefaultLineLineWidth',1);
        set(gcf,'DefaultAxesFontSize',14);
        hold on;
        grid on;
        xlabel('Datarate bruker 1 (kort), Mbps');
        ylabel('Datarate bruker 2 (lang), Mbps');
        plot(rate1,rate2);
        plot(ratebruker1,ratebruker2,'rx');
        legend('OSM: Dual Dec.','Punkter funnet','location','northeast')
        axis(akseverdier);
        title(tittel);
    end
elseif run == 3
    exhaustive;
    ex1 = rate1verdier;
    ex2 = rate2verdier;
    dualdec;
    dd1 = rate1;
    dd2 = rate2;
    stopp_tid = clock;
    if plotting == 1
        set(gcf,'DefaultLineLineWidth',1)   %  Default line size
        set(gcf,'DefaultAxesFontSize',14)   %  Default axis font size
        hold on;
        grid on;
        xlabel('Datarate bruker 1 (kort), Mbps');
        ylabel('Datarate bruker 2 (lang), Mbps');
        plot(dd1,dd2);
        plot(ex1,ex2,'--');
        legend('OSM: Dual Dec.','OSM: Ex.search','location','northeast')
        axis(akseverdier);
        title(tittel);
    end
end
brukt_tid = stopp_tid - start_tid;
brukt_tid_sekunder = brukt_tid(5)*60 + brukt_tid(6)
```

6.2 exhaustive.m

exhaustive.m er filen som kjører Exhaustive search og beregner rateregionen for dette. Den benytter exfuntion.m til beregningene.

```
global F antallbit pknm skn NF sendt ratebruker1 ratebruker2 rateteller
sigmakn gamma

%% Initalisering
L = [ 0.5 0.8 ];
NU = length(L);
L_o = min(L);
```

```
NF = length(F);
B = 4.3125*10^-3;
Bupstream = 5.2-3.75;
St = -52;
WN = 140;
fext1dB = 45;
Margin = 5;

%% Koeffisientutregning
alpha = 22.5*log(10)/20.*sqrt(F);
gamma = 10^(Margin/10);
sendt = 10^(St/10);
w_kn = 10^(-WN/10);

%% Beregninger
for k=1:NF
    for n=1:NU
        akn(k,n) = exp(-2*alpha(k)*L(n));
        sigmakn(k,n) = w_kn;
        for m=1:NU
            pknm(k,n,m) = exp(-2*alpha(k)*L(m))*10^(-fext1dB/10)*F(k)^2*L_o;
        end
    end
end
for k=1:NF
    for n=1:NU
        for m=1:NU
            for bn=0:antallbit
                for bm=0:antallbit
                    teller(k,n,bn+1,bm+1) = (2^bn-1)*gamma*((2^bm-
1)*gamma*pknm(k,n,m)*sigmakn(k,m) + akn(k,m)*sigmakn(k,n));
                    nevner(k,n,bn+1,bm+1) = akn(k,n)*akn(k,n)-(2^bn-1)*(2^bm-
1)*gamma^2*pknm(k,n,m)*pknm(k,n,m);
                    skn(k,n,bn+1,bm+1)    =
(teller(k,n,bn+1,bm+1))/(nevner(k,n,bn+1,bm+1));
                end
            end
        end
    end
end

%% Exhaustive search
ratebruker1 = 0;
ratebruker2 = 0;
tarmslyng = 0;
rateteller = 0;
exfunction(1,NU*NF);

%% Finne rateregionen
denfinnes=0;
verditeller=1;
verdivektor=0;
rate1verdier = -1;
for temp1=1:rateteller
    for temp2=1:length(rate1verdier)
        if ratebruker1(temp1)==rate1verdier(temp2)
            denfinnes=1;
        end
    end
    if denfinnes==0
        rate1verdier(verditeller)=ratebruker1(temp1);
        verdivektor(verditeller)=temp1;
        verditeller=verditeller+1;
    end
    denfinnes=0;
end
rate1verdier = sort(rate1verdier);
rate2verdier = zeros(1,length(rate1verdier))-1;
```

```
for temp3=1:rateteller
    for temp4=1:length(rate1verdier)
        if rate1verdier(temp4)==ratebruker1(temp3)
            if ratebruker2(temp3)>rate2verdier(temp4)
                rate2verdier(temp4)=ratebruker2(temp3);
            end
        end
    end
end
rate1verdier(length(rate1verdier)+1)=rate1verdier(length(rate1verdier));
rate2verdier(length(rate2verdier)+1)=0;
for temp4=1:length(rate2verdier)-1
    if rate2verdier(length(rate2verdier)-
temp4)<rate2verdier(length(rate2verdier)+1-temp4)
        rate2verdier(length(rate2verdier)-
temp4)=rate2verdier(length(rate2verdier)+1-temp4);
        rate1verdier(length(rate2verdier)-
temp4)=rate1verdier(length(rate2verdier)+1-temp4);
    end
end
```

6.3 exfunction.m

exfunction.m er funksjonsfilen som kjører seg selv så mange ganger som det er nødvendig avhengig av antall brukere og antall delbånd. Til slutt beregner den tilhørende verdier.

```
function exfunction(nivaa,total)

global antallbit bitvektor pknm skn NF sendt ratebruker1 ratebruker2 rateteller
sigmakn gamma

if nivaa == total+1
    totaleffektbruker1 = 0;
    totaleffektbruker2 = 0;
    for k=1:NF
        totaleffektbruker1 = totaleffektbruker1 + skn(k,1,bitvektor(2*k-
1),bitvektor(2*k));
        totaleffektbruker2 = totaleffektbruker2 +
skn(k,2,bitvektor(2*k),bitvektor(2*k-1));
    end
    if totaleffektbruker1<=sendt*NF
        if totaleffektbruker2<=sendt*NF
            rateteller = rateteller + 1;
            rater = zeros(2,k);
            for k=1:NF
                fextsum = pknm(k,1,2)*skn(k,1,bitvektor(2*k-1),bitvektor(2*k));
                rater(1,k) = floor(
log2(1+1/gamma*skn(k,2,bitvektor(2*k),bitvektor(2*k-1))/(fextsum+sigmakn(1,1))) );
                fextsum = pknm(k,2,1)*skn(k,2,bitvektor(2*k),bitvektor(2*k-1));
                rater(2,k) = floor( log2(1+1/gamma*skn(k,1,bitvektor(2*k-
1),bitvektor(2*k))/(fextsum+sigmakn(1,2))) );
            end
            ratebruker1(rateteller) = sum ( rater(1,:) );
            ratebruker2(rateteller) = sum ( rater(2,:) );
        end
    end
else
    for i=1:antallbit+1
        bitvektor(nivaa) = i;
        exfunction(nivaa+1,total);
    end
end

end
```

6.4 dualdec.m

dualdec.m er filen som kjører Dual decomposition og finner rateregionen ut i fra dette. Se for øvrig Figur 1.4.5.

```
global NF F antallbit gamma pknm skn sigmakn P1 P2;

%% Initalisering
L = [ 0.5 0.8 ];
NU = length(L);
L_o = min(L);
B = 4.3125*10^-3;
NF = length(F);
Bupstream = 5.2-3.75;
St = -52;
WN = 140;
fext1dB = 45;
Margin = 5;

%% Koeffisientutregning
alpha = 22.5*log(10)/20.*sqrt(F);
gamma = 10^(Margin/10);
sendt = 10^(St/10);
w_kn = 10^(-WN/10);
P1 = sendt*NF;
P2 = sendt*NF;

%% Beregninger
for k=1:NF
    for n=1:NU
        akn(k,n) = exp(-2*alpha(k)*L(n));
        sigmakn(k,n) = w_kn;
        for m=1:NU
            pknm(k,n,m) = exp(-2*alpha(k)*L(m))*10^(-fext1dB/10)*F(k)^2*L_o;
        end
    end
end
for k=1:NF
    for n=1:NU
        for m=1:NU
            for bn=0:antallbit
                for bm=0:antallbit
                    teller(k,n,bn+1,bm+1)=(2^bn-1)*gamma*((2^bm-
1)*gamma*pknm(k,n,m)*sigmakn(k,m) + akn(k,m)*sigmakn(k,n));
                    nevner(k,n,bn+1,bm+1)=akn(k,n)*akn(k,n)-(2^bn-1)*(2^bm-
1)*gamma^2*pknm(k,n,m)*pknm(k,n,m);

skn(k,n,bn+1,bm+1)=(teller(k,n,bn+1,bm+1))/(nevner(k,n,bn+1,bm+1));
                end
            end
        end
    end
end

%% Dual Decomposition
rateteller=0;
for w = 0:0.01:1
    [s1,s2] = optimize_lambda1(w);
    rateteller = rateteller + 1;
    fextsum = pknm(1,1,2)*s2(1);
    rates(1,1) = floor( log2(1+1/gamma*s1(1)/(fextsum+sigmakn(1,1))) );
    fextsum = pknm(1,2,1)*s1(1);
    rates(1,2) = floor( log2(1+1/gamma*s2(1)/(fextsum+sigmakn(1,2))) );
    fextsum = pknm(2,1,2)*s2(2);
    rates(2,1) = floor( log2(1+1/gamma*s1(2)/(fextsum+sigmakn(2,1))) );
    fextsum = pknm(2,2,1)*s1(2);
    rates(2,2) = floor( log2(1+1/gamma*s2(2)/(fextsum+sigmakn(2,2))) );
```

```
    fextsum = pknm(3,1,2)*s2(3);
    rates(3,1) = floor( log2(1+1/gamma*s1(3)/(fextsum+sigmakn(3,1))) );
    fextsum = pknm(3,2,1)*s1(3);
    rates(3,2) = floor( log2(1+1/gamma*s2(3)/(fextsum+sigmakn(3,2))) );
    ratebruker1(rateteller) = sum ( rates(:,2) );
    ratebruker2(rateteller) = sum ( rates(:,1) );
end

%% Finne rateregionen
denfinnes=0;
verditeller=1;
verdivektor=0;
rate1verdier = -1;
for temp1=1:rateteller
    for temp2=1:length(rate1verdier)
        if ratebruker1(temp1)==rate1verdier(temp2)
            denfinnes=1;
        end
    end
    if denfinnes==0
        rate1verdier(verditeller)=ratebruker1(temp1);
        verdivektor(verditeller)=temp1;
        verditeller=verditeller+1;
    end
    denfinnes=0;
end
rate1verdier = sort(rate1verdier);
rate2verdier = zeros(1,length(rate1verdier))-1;
for temp3=1:rateteller
    for temp4=1:length(rate1verdier)
        if rate1verdier(temp4)==ratebruker1(temp3)
            if ratebruker2(temp3)>rate2verdier(temp4)
                rate2verdier(temp4)=ratebruker2(temp3);
            end
        end
    end
end
rate1verdier(length(rate1verdier)+1)=rate1verdier(length(rate1verdier));
rate2verdier(length(rate2verdier)+1)=0;
for temp4=1:length(rate2verdier)-1
    if rate2verdier(length(rate2verdier)-
temp4)<rate2verdier(length(rate2verdier)+1-temp4)
        rate2verdier(length(rate2verdier)-
temp4)=rate2verdier(length(rate2verdier)+1-temp4);
        rate1verdier(length(rate2verdier)-
temp4)=rate1verdier(length(rate2verdier)+1-temp4);
    end
end
rate1(1) = 0;
rate2(1) = rate2verdier(1);
for temp5=1:length(rate2verdier)
    rate1(temp5+1)=rate1verdier(temp5);
    rate2(temp5+1)=rate2verdier(temp5);
end
```

6.5 optimize_lambda1.m

Se Figur 1.4.5.

```
function [s1,s2] = optimize_lambda1(w)

global P1 epsilon;

lambda1max = 1;
lambda1min = 0;
s1=0;
s2=0;
```

```
while sum(s1(:)) > P1
    lambda1max = 2 * lambda1max;
    [s1,s2] = optimize_lambda2(w,lambda1max);
end
differanse = 1;
while differanse > epsilon
    lambda1 = (lambda1max + lambda1min) / 2;
    [s1,s2] = optimize_lambda2(w,lambda1);
    if sum(s1(:)) > P1
        lambda1min = lambda1;
    else
        lambda1max = lambda1;
    end
    differanse = abs(lambda1max - lambda1min);
end

end
```

6.6 optimize_lambda2.m

Se Figur 1.4.5.

```
function [s1,s2] = optimize_lambda2(w,lambda1)

global P2 epsilon;

lambda2max = 1;
lambda2min = 0;
s1=0;
s2=0;
while sum(s2(:)) > P2
    lambda2max = 2 * lambda2max;
    [s1,s2] = optimize_s(w,lambda1,lambda2max)
end
differanse = 1;
while differanse > epsilon
    lambda2 = (lambda2max + lambda2min) / 2;
    [s1,s2] = optimize_s(w,lambda1,lambda2);
    if sum(s2(:)) > P2
        lambda2min = lambda2;
    else
        lambda2max = lambda2;
    end
    differanse = abs(lambda2max - lambda2min);
end

end
```

6.7 optimize_s.m

Se Figur 1.4.5.

```
function [s1,s2] = optimize_s(w,lambda1,lambda2)

global NF antallbit gamma pknm skn sigmakn;

for k=1:NF
    L(k)=0;
    for bitbruker1=1:antallbit
        for bitbruker2=1:antallbit
            fext1 = pknm(k,1,2)*skn(k,2,bitbruker2,bitbruker1);
            bk1 = floor(
log2(1+1/gamma*skn(k,1,bitbruker1,bitbruker2)/(fext1+sigmakn(k,1))) );
            fext2 = pknm(k,2,1)*skn(k,1,bitbruker1,bitbruker2);
            bk2 = floor(
log2(1+1/gamma*skn(k,2,bitbruker2,bitbruker1)/(fext2+sigmakn(k,2))) );
```

```
            Ltemp = w*bk1 + (1-w)*bk2 - lambda1*skn(k,1,bitbruker1,bitbruker2) -
lambda2*skn(k,2,bitbruker2,bitbruker1);
            if L(k) < Ltemp
                L(k) = Ltemp;
                s1temp = skn(k,1,bitbruker1,bitbruker2);
                s2temp = skn(k,2,bitbruker1,bitbruker2);
            end
        end
    end
    s1(k) = s1temp;
    s2(k) = s2temp;
end

end
```

6.8 bit_time_plot.m

bit_time_plot.m plotter beregningstiden med hensyn på bitloadingen. Tidene er de som er beregnet av run_plot.m.

```
clear all;

%% Initialisering
run = 3; % EX:1, DD:2, EX & DD:3

%% Plot
if run == 1
    bitload = [3 5 7];
    extid = [0.6880 36.9530 1475.4220];
elseif run == 2
    bitload = [3 5 7 10 12 15];
    ddtid = [14.7180 20.3280 28.4530 45.7180 150.8120 231.9220];
elseif run == 3
    bitload = [3 5 7];
    ddtid = [14.7180 20.3280 28.4530];
    extid = [0.6880 36.9530 1475.4220];
end
set(gcf,'DefaultLineLineWidth',1);
set(gcf,'DefaultAxesFontSize',14);
hold on;
grid on;
xlabel('Bitloading [bit/s/Hz/carrier]');
ylabel('Beregningstid [s]');
if run == 1
    plot (bitload,extid,'--');
    legend('Exhaustive search','location','northwest')
    axis([3 7 -100 1500]);
elseif run == 2
    plot (bitload,ddtid);
    legend('Dual decomposition','location','northwest')
    axis([3 15 0 250]);
elseif run == 3
    plot (bitload,extid,'--');
    plot (bitload,ddtid);
    legend('Exhaustive search','Dual decomposition','location','northwest');
    axis([3 7 -100 1500]);
end
title('Bitload/Tid');
```

6.9 freq_time_plot.m

bit_time_plot.m plotter beregningstiden med hensyn på antall delbånd for Dual decomposition. Tidene er de som er beregnet av run_plot.m.

```
clear all;
```

```
%% Plot

delband = [3 4 5 6 7];
ddtid = [14.7180 17.3590 19.6250 22.3430 25.1250];
set(gcf,'DefaultLineLineWidth',1);
set(gcf,'DefaultAxesFontSize',14);
hold on;
grid on;
xlabel('Antall delbånd');
ylabel('Beregningstid [s]');
plot (delband,ddtid);
legend('Dual decomposition','location','northwest')
title('Delbånd/Tid');
```