Katsaus LDPC-koodeihin sekä niiden rooliin 5G- ja 6G-verkoissa

Turun yliopisto
Tietotekniikan laitos
TkK-tutkielma
Tietotekniikka
Kesäkuu 2026
Valtteri Collin

TURUN YLIOPISTO
Tietotekniikan laitos

Valtteri Collin: Katsaus LDPC-koodeihin sekä niiden rooliin 5G- ja 6G-verkoissa

TkK-tutkielma, 29 s.
Tietotekniikka
Kesäkuu 2026

Virheenkorjauskoodit ovat oleellinen osa takaamaan langattoman tietoliikenteen luotettavuutta. Robert R. Gallagerin 1960-luvulla kehittämät LDPC-koodit ovat sittemmin saaneet lisää huomioita muiden tutkijoiden, kuten R. Tannerin teosten kautta ja ovat nykyään osana 5G-standardia. LDPC-koodien teoreettiset ominaisuudet mahdollistavat niiden tehokkaan dekoodauksen iteratiivisilla Message-Passing-algoritmeilla, mikä on niiden oleellisin ero perinteisiin lohkokoodeihin. Bit-Flipping-, Sum-Product ja Min-Sum-algoritmit, jotka kuuluvat kyseiseen algoritmiperheeseen esitellään. Lisäksi esitellään yleinen koodausalgoritmi, joka kykenee lineaariaikaiseen koodaukseen.
LDPC-koodit on onnistuneesti implementoitu 5G:hen, ja ne ovat vastuussa datalähetysten virheenkorjauksesta. Polar-koodit ovat myös käytössä 5G:ssä ohjauskanavilla. Kirjallisuudessa esitetyt tulokset paljastavat, että edellisen sukupolven Turbo-koodeihin verrattuna, tämä kokoonpano toimii tehokkaammin laajemmalla määrällä eri lohkonpituuksia.
6G-verkkojen oletetaan kattavan yhä enemmän käyttökohteita nykysukupolveen verrattuna, ja suorituskykyä tullaan kasvattamaan esimerkiksi siirtonopeuden ja latenssin osalta. Tästä johtuen vaatimukset ovat suuret, ja niitä on enemmän. Vaikka kirjallisuudessa on esitetty muitakin lähestymistapoja, ovat LDPC-, Polar- ja Turbo-koodit edelleen varteenotettavia menetelmiä seuraavan sukupolven kanavakoudasta varten.

Asiasanat: LDPC-koodit, Polar-koodit, Turbo-koodit, Parillisuusmatriisi, Tanner-graafi, Koodaus, Dekoodaus, Message-passing, Bit-Flipping, Sum-Product, Min-Sum, 5G, 6G, KPI

Sisällys

1 Johdanto

Tietoliikennekommunikaation perustavanlaatuisin ongelma on se, kuinka tietoa voidaan lähettää tehokkaasti ja luotettavasti virhealttiin kanavan ylitse. Koodausteoria on tietotekniikan ala, joka pyrkii vastaamaan tähän kysymykseen ja kehittämään menetelmiä, joilla virheitä tiedonlähetyksessä voidaan havainnoida sekä korjata. Näitä menetelmiä kutsutaan virheenkorjauskoodeiksi.

Eräitä tällaisia virheenkorjauskoodeja ovat Low-Density Parity-Check (LDPC) -koodit. LDPC-koodit ovat 1960-luvulla Robert R. Gallagerin kehittämä virheenkorjaava lohkokoodi [1]. Gallagerin aikaisten tietoliikennejärjestelmien laskentateho ei kuitenkaan riittänyt hänen innovaationsa hyödyntämiseen. Teknologian kehityksen myötä kiinnostus LDPC-koodeja kohtaan heräsi uudelleen 1990-luvulla. Lisäksi kiinnostusta LDPC-koodeja kohtaan herätti R. Tannerin merkittävä työ "A Recursive Approach to Low Complexity Codes", jossa Tanner esittelee LDPC-koodeille graafisen esitystavan, eli niin kutsutun Tanner-graafin [2].

Eräs oleellisimmista LDPC-koodien sovelluskohteista on tänä päivänä niiden käyttö 5G-mobiiliverkkojen kanavakoodausmenetelmänä, jossa ne ovat osana takaamassa modernien langattomien teknologioiden luotettavuutta. Siirtymä näissä mobiiliverkoissa sukupolvesta toiseen on asettanut joka kerta merkittäviä teknologisia vaatimuksia, eikä seuraava sukupolvenvaihdos ole poikkeus. LDPC-koodien onnistunut implementaatio nykyisiin langattomien verkkojen teknologioihin on antanut niille luotettavan maineen. Tämä luottamus puolestaan on pitänyt kiinnostusta me-

netelmää kohtaan yllä alan tutkimuskentällä 6G:n kontekstissa.

Työssä on neljä lukua. Luvussa 2 tarkastellaan LDPC-koodeja teoreettisesta näkökulmasta, ja näin ollen selvitetään lukijalle mitä LDPC-koodit ovat, millaisia ominaisuuksia niillä on ja millaisia algoritmeja niiden koodauksessa ja dekoodauksessa käytetään. Luvussa 3 siirrytän taas käsittelemään aihetta käytännön näkökulmasta. LDPC-koodien asemaan kanavakoodausmenetelmänä 5G:ssä tutustutaan tarkemmin. Lisäksi käsittelyyn otetaan mukaan kaksi muuta nykyiselle koodausteorialle, ja langattomalle verkkoliikenteelle olennaista menetelmää: Polar-koodit, jotka ovat myös käytössä 5G-verkoissa, ja Turbo-koodit, jotka olivat kahden edellisen sukupolven päätoiminen virheenkorjausmenetelmä. Lopuksi käsitellään vielä 6G-siirtymää, ja tähän liittyviä haasteita kanavakoodauksen kontekstissa.

Tutkielman alkupuoliskon teoriakatsauksen tukemana luvussa 3 pyritään vastaamaan tutkimuskysymyksiin:

TK 1: Miten LDPC-koodien suorituskyky vertautuu muihin vallitseviin menetelmiin? Luvussa 3.2 vertaillaan edellä mainittuja virheenkorjausmenetelmiä keskenään ja tehdään johtopäätöksiä vertailun tuloksista.

TK 2: Millaisia haasteita 6G-siirtymä asettaa kanavakoodaukselle, ja riittävätkö nykyiset menetelmät vastaamaan niihin? Luvussa 3.3 tutkitaan 6G:n haasteita kanavakoodauksen ja KPI (Key Performance Indicator) tavoitteiden kontekstissa. Lopuksi arvioidaan aiemmin käsiteltyjen koodausmenetelmien potentiaalia näiden haasteiden ratkaisijoina.

Tutkielma on kirjallisuuskatsaus. Käsittelyluvun 3 materiaali on haettu lähes yksinomaan IEEE:n tietokannasta muutamaa artikkelia lukuun ottamatta. 3GPP:n 5G-standardi ja IIS-raportti haettiin suoraan organisaatioiden sivuilta. Käytetyt hakusanat olivat: LDPC, Turbo, Polar, Error correction, Channel coding, Performance,

Review, 5G, 6G. Näistä rakennettiin muutama eri yhdistelmä, joilla haut tietokantaan toteutettiin. Tarkkaa rajaa julkaisuajankohdalle ei asetettu, mutta mahdollisimman tuoreita julkaisuja suosittiin. Vanhin julkaisuvuosi aineistossa on 2014. Tämän lisäksi tärkein sisäänluvun kriteeri oli relevanssi tutkimuskysymyksiin nähden. Menetelmien vertailuun liittyvät artikkelit pyrittiin valitsemaan niin, että LDPC-, Polar- ja Turbo-koodit olivat niiden pääkohteena.

Luvun 2 materiaalin sisäänluvulle kriteerit olivat vapaammat. Materiaali on haettu IEEE:n tietokannasta, mutta myös muun muassa google scholarista ja ResearchGatesta. Tärkeintä oli, että lähteissä esitetty teoria oli keskenään yhdenmukaista. Lisäksi alkuperäisiä teoksia pyrittiin suosimaan, missä mahdollista.

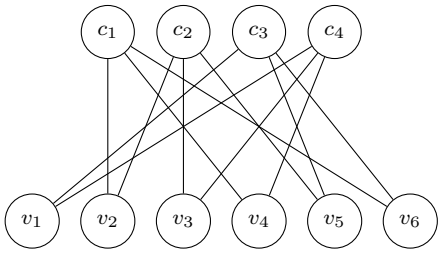
2 LDPC-koodit

Robert. R. Gallager kuvailee LDPC-koodit esittelevässä työssään käsitteen seuraavasti: LDPC-koodit ovat eräänlaisia parillisuuskoodeja. LDPC-koodin määrittää parillisuusmatriisi H, joka koostuu pääosin nollista, ja pienestä määrästä ykkösiä. Tarkemmin (n, j, k) LDPC-koodi on koodi, jonka pituus on n, ja jonka matriisin jokainen sarake sisältää j ykköstä, ja jokainen rivi k ykköstä [1]. Tämänlaista LDPC-koodia, jossa siis H:n jokaisen rivin ja sarakkeen ykkösten määrä on vakio, kutsutaan myös säännölliseksi LDPC-koodiksi. Vastoin, jos ykkösten rivi- ja sarakekohtainen määrä ei ole vakio, kutsutaan LDPC-koodia epäsäännölliseksi. [3].

LDPC-koodien graafinen esitys Tanner-graafina, tai joskus myös niin kutsuttuna tekijägraafina (Factor graph) antaa yhtä täydellisen esitystavan LDPC-koodeille kuin perinteinen parillisuusmatriisi [3][4]. Kirja "Modern coding theory" määrittelee lyhyesti LDPC-koodit koodeiksi, joilla on vähintään yksi harva Tanner-graafi [5]. Se että graafi, ja näin myös matriisi, on harva, on myös oleellinen ominaisuus LDPC-koodien purkamisessa käytettyjen algoritmien tehokkuuden kannalta [6]. Tanner-graafit antavat erityisen edun LDPC-koodeja purkavien algoritmien suunnittelussa ja visualisoinnissa, minkä takia ne ovat vakiintuneet yhtä yleiseksi esitystavaksi kuin matriisiesitykset. Suurin ero LDPC-koodien ja klassisten lohkokoodien välillä onkin niiden purkamisessa käytetyt menetelmät. LDPC-koodeja puretaan iteratiivisin menetelmin niiden Tanner-graafia hyödyntäen, kun taas klassisten blokkikoodien kohdalla tavallisesti käytetty menetelmä on Maximum likelihood decoding [7].

Parillisuusmatriisia H vastaava Tanner-graafi on kaksijakoinen graafi (bipartite graph), joka koostuu muuttujasolmuista ja tarkastussolmuista. Muuttujasolmuja on n kappaletta, eli yksi per koodisanan komponentti. Tarkastussolmuja taas on yksi per parillisuusyhtälö, joka on siis myös H:n rivien määrä m. Tarkastussolmun j ja muuttujasolmun i välille piirretään kaari, jos $h_{j,i} = 1$, eli toisin sanoen, jos se osallistuu j:nnenteen parillisuusyhtälöön. Vastaavasti H voidaan määrittää Tanner-graafin kautta asettamalla $h_{j,i} = 1$, jos j:nnennen tarkastussolmun ja i:nnennen muuttujasolmun välillä on kaari [5][3].

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



Kuva 2.1: $(6, 2, 3)$ LDPC parillisuusmatriisi (vas), ja sitä vastaava Tanner-graafi (oik), jossa v_i on muuttujasolmu ja c_i tarkastussolmu.

Kuvan 2.1 parillisuusmatriisin ja sen Tanner-graafin muodostama LDPC-koodi on säännöllinen. Sen sarakepaino $w_c = 2$ ja rivipaino $w_r = 3$, jotka myös vastaavat Tanner-graafin muuttuja- ja tarkastussolmujen asteita, eli siis niihin yhdistyvien kaarien määrää. Koska epäsäännöllisillä LDPC-koodeilla nämä arvot eivät ole vakioita, käytetään niiden sijasta tavallisesti astejakaumapolynomeja (Degree distribution) merkitsemään kuinka suuri osa kaarista yhdistyy kunkin asteisiin solmuihin. Muuttujasolmuille astejakaumapolynomi on

$$\lambda(x) = \sum_{d=1}^{d_v} \lambda_d x^{d-1},$$

jossa λ_d merkitsee niiden kaarien osamäärää graafin kaikkiin kaariin nähden, jotka yhdistyvät astetta d olevaan muuttujasolmuun. d_v puolestaan merkitsee muuttujasolmujen suurinta astetta. Samankaltaisesti tarkastussolmujen astejakaumapolyno-

mi on

$$\rho(x) = \sum_{d=1}^{d_c} \rho_d x^{d-1},$$

jossa ρ_d merkitsee niiden kaarien määrää graafin kaikkiin kaariin nähden, jotka yhdistyvät astetta d olevaan tarkastussolmuun, ja d_c puolestaan merkitsee tarkastussolmujen suurinta astetta. Kuvan 2.1 määrittämälle koodille on siis $\lambda(x) = x$ ja $\rho(x) = x^2$ [3].

Tanner-graafiin muodustuvat syklit ovat myös oleellinen LDPC-koodien ominaisuus. Syklit ovat graafin kaarista muodostuvia polkuja, jotka kiertävät takaisin solmuun, josta polku alkoi, käyttämättä yhtään kaarta enempää kuin kerran. Lyhyillä, etenkin neljän kaaren mittaisilla sykleillä on häiritsevä vaikutus tarkastussolmujen astejakaumaan ja ne heikentävät LDPC-koodien purkuun käytettävien iteratiivisten dekoodausalgoritmien suorituskykyä. Tarkalleen ottaen lyhyet syklit voivat estää algoritmia konvergoitumasta. Pitkille koodeille parillisuusmatriisi H on erittäin harva, joten syklejä esiintyy myös vähemmän ja niistä koituvat vaikutukset ovat lievemmät. Lyhyempien LDPC-koodien suunnittelussa etenkin neljän kaaren mittaisia syklejä pyritään kuitenkin välttämään [3][8]. Yleisesti ottaen, mitä pidempiä LDPC-koodit ovat, ja mitä epäsäännöllisempiä niiden astejakaumat ovat, sitä paremman suorituskyvyn ne takaavat. Kanavan kapasiteettia lähenevät LDPC-koodit ovat hyvin pitkiä ja epäsäännöllisiä. Tunnetuin esimerkki tästä on Sae-Young Chungin väitöskirjaan perustuvassa tutkimuksessa, jossa tarkasteltujen koodien pituus oli 10^7 ja muuttujasolmujen asteet vaihtelivat välillä 20-8000. Simulaatioissa saavutettiin AWGN-kanavan (Additive White Gaussian Noise) kapasiteetti 0.04 dB:n etäisyydellä bittivirhesuhteella 10^{-6} (Bit Error Rate) [9].

LDPC-koodeille on kehitetty useita konstruointimetodeja, joista kukin pyrkii tyydyttämään tiettyjä kriteerejä, kuten tehokas koodaus ja dekoodaus tai suorituskyky mahdollisimman lähellä kanavan kapasiteettia.

Suoraviivainen esimerkki tällaisesta on Gallagerin alkuperäinen menetelmä. Gal-

lagerin LDPC-koodit ovat säännöllisiä (n, j, k) LDPC-koodeja, joiden parillisuusmatriisi H jaetaan j osaan niin, että jokaisen alamatriisin sarake sisältää yhden ykkösen. H:n ylimmässä alamatriisissa ykköset ovat laskevassa järjestyksessä niin, että i:nnellä rivillä ykköset ulottuvat sarakkeelta $(i-1)k+1$ sarakkeelle ik. Loput alamatriisit ovat sarakepermutaatioita ensimmäisestä [1].

$$H = \begin{pmatrix} H_1 \\ H_2 \\ \vdots \\ H_j \end{pmatrix} \qquad H_1 = \begin{pmatrix} 1 & 1 & 0 & 0 & 0 & 0 & & & & & & \\ 0 & 0 & 1 & 1 & 0 & 0 & \cdots & & & \cdots & & \\ 0 & 0 & 0 & 0 & 1 & 1 & & & & & & \\ & & \vdots & & & & \ddots & & & \vdots & & \\ & & & & & & & 1 & 1 & 0 & 0 & 0 & 0 \\ & & \cdots & & & & \cdots & 0 & 0 & 1 & 1 & 0 & 0 \\ & & & & & & & 0 & 0 & 0 & 0 & 1 & 1 \end{pmatrix}$$

Kuva 2.2: Gallagerin menetelmän mukaan rakennettu matriisi H (vas), ja sen ylin alamatriisi H_1 (oik).

Yleisesti ottaen yksittäisen pitkän koodin tarkastelun sijaan on helpompaa tutkia kaikkien (n, j, k) koodien joukkoa koodisanojen suuren määrän vuoksi. Yksinkertaisuudestaan huolimatta tähän menetelmään perustuvilla koodijoukoilla on erinomainen minimietäisyys, joka kasvaa lineaarisesti koodin pituuden n suhteen. Tämä menetelmä ei kuitenkaan takaa, että koodi olisi vapaa lyhyistä sykleistä [1].

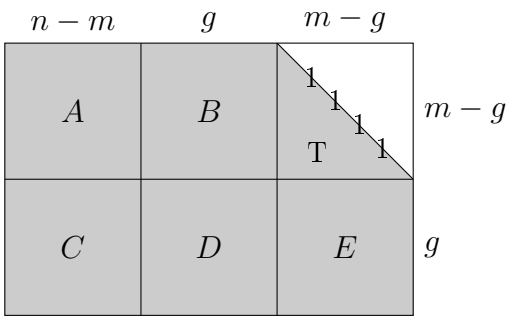
2.1 Koodaus

LDPC-koodit ovat siitä erikoisia, että niiden koodaus ei ole itsestäänselvä osa niiden määritelmää. Eräs kritiikki LDPC-koodeja kohtaan onkin niiden näennäisesti monimutkainen koodausprosessi [5][10]. T.J. Richardson ja R. Urbanke näyttävät kuitenkin, että LDPC-koodeja voidaan koodata lineaariajassa, ja että tapauksissa joissa aikakompleksisuus on neliöllinen, on tämän termin kerroin pieni [10]. Kirja "Modern coding theory" esittää samoihin periaatteisiin perustuvan algoritmin kuin T.J. Richardson ja R. Urbanke, kuvaillen sitä yleiseksi LDPC-koodien koodausme-

netelmäksi [5]. Lineaariaikaisen koodauksen saavuttaminen ei kuitenkaan ole mahdollista yleisesti, vaan vaatii LDPC-koodin optimointia tätä varten [5][10]. Tämän aliluvun tarkoituksena on esitellä kyseinen algoritmi.

Algoritmin tehokkuus perustuu LDPC-koodien oleellisimpaan ominaisuuteen, eli niiden harvaan parillisuusmatriisiin H. Määritelmän mukaan $m \times n$-parillisuusmatriisin H koodi, koostuu kaikista n-pituisista vektoreista $\boldsymbol{x}$, jotka toteuttavat yhtälön $H\boldsymbol{x}^T = \mathbf{0}^T$. Yksinkertaisin tapa rakentaa koodausalgoritmi tällaiselle koodille, on saattaa H ala- tai yläkolmiomuotoon Gauss -Jordan menetelmällä, ja jakaa $\boldsymbol{x}$ informaatio- ja parillisuusosiin $\boldsymbol{x} = (\boldsymbol{s}, \boldsymbol{p})$, jossa $\boldsymbol{s} \in \mathbb{F}_2^{n-m}$ ja $\boldsymbol{p} \in \mathbb{F}_2^m$. Tämän jälkeen $\boldsymbol{s}$ voidaan täyttää halutuilla informaatiobiteillä, ja $\boldsymbol{p}$ määrittää takaisinsijoituksella. Kyseinen metodi ei kuitenkaan saavuta lineaariaikaista koodausta, sillä H:n esikäsittely Gauss -Jordan menetelmällä vaatii $O(n^3)$ operaatioita, ja varsinainen koodaus $O(n^2)$ operaatioita. Tämä johtuu siitä, että yleisesti ottaen H ei ole enää harva esikäsittelyn jälkeen [10].

T.J.Richardsonin ja R. Urbanken koodausalgoritmin tavoitteena on saattaa H allaolevaan, niin kutsuttuun likimääräiseen alakolmiomuotoon ainoastaan permutaatioiden avulla. Tällöin H pysyy edelleen harvana [10].



Alamatriisin (CDE) korkeutta g kutsutaan väliksi (gap), ja se mittaa H:n etäisyyttä täydellisestä alakolmiomuodosta. T.J.Richardson ja R. Urbanke osoittavat, että tällaisen matriisin koodauksen aikakompleksisuus on ylhäältä rajoitettu termillä $n + g^2$, ja että jos T:n dimensiot ovat $(m-g) \times (m-g)$, niin koodaus on mahdollis-

ta suorittaa lineaariajassa. Kyseistä lähestymistapaa käyttävät algoritmit pyrkivät luonnollisesti siis minimoimaan g:n arvoa suorituskyvyn parantamiseksi [10][5].

Kun H on saatettu approksimaattiseen alakolmiomuotoon, seuraava askel on nollata alamatriisi E tulolla

$$\begin{pmatrix} I & 0 \\ -ET^{-1} & I \end{pmatrix} \begin{pmatrix} A & B & T \\ C & D & E \end{pmatrix} = \begin{pmatrix} A & B & T \\ -ET^{-1}A + C & -ET^{-1}B + D & 0 \end{pmatrix}$$

ja jakaa parillisuusvektori $\boldsymbol{p}$ kahteen osaan niin, että $\boldsymbol{x} = (\boldsymbol{s}, \boldsymbol{p_1}, \boldsymbol{p_2})$, jossa $\boldsymbol{p_1}$:n pituus on g, ja $\boldsymbol{p_2}$:n pituus on $m - g$. Nyt $H\boldsymbol{x}^T = \boldsymbol{0}^T$ jakaantuu kahteen yhtälöön.

$$A\boldsymbol{s}^T + B{\boldsymbol{p_1}}^T + T{\boldsymbol{p_2}}^T = \boldsymbol{0}^T, ja$$
$$(-ET^{-1}A + C)\boldsymbol{s}^T + (-ET^{-1}B + D){\boldsymbol{p_1}}^T = \boldsymbol{0}^T.$$

Määritellään

$$\phi = -ET^{-1}B + D$$

ja oletetaan, että se on kääntyvä. Tällöin ${\boldsymbol{p_1}}^T = -\phi^{-1}(-ET^{-1}A + C)\boldsymbol{s}^T$. Nyt, jos $g \times (n-m)$ kokoinen matriisi $-\phi^{-1}(-ET^{-1}A + C)$ lasketaan etukäteen, $\boldsymbol{p_1}$ voidaan määrittää $O(g \times (n-m))$ ajassa kertomalla $\boldsymbol{s}^T$ tällä matriisilla. Jäljelle jäävä parillisuusvektori $\boldsymbol{p_2}$ ratkaistaan puolestaan yhtälöstä ${\boldsymbol{p_2}}^T = -T^{-1}(A\boldsymbol{s}^T + B{\boldsymbol{p_1}}^T)$. Molempien parillisuusvektorien määrittämistä voidaan tehostaa jakamalla laskut pienempiin osiin, jolloin $\boldsymbol{p_1}$ voidaan loppujen lopuksi ratkaista $O(n + g^2)$ ajassa, ja $\boldsymbol{p_2}$ $O(n)$ ajassa [10][5].

Esilaskennassa H saatetaan ensin likimääräiseen alakolmiomuotoon rivi- ja sarakepermutaatioiden avulla, niin että väli g on mahdollisimman pieni. Tämän jälkeen alamatriisi E nollataan aiemmin mainitulla matriisitulolla, tai tässä tapauksessa tehokkaammin Gauss-Jordan-menetelmällä. Jos tuloksena saatu matriisi ϕ on kääntyvä, on esilaskenta valmis. Jos taas ϕ ei ole kääntyvä, voidaan siitä tehdä sellainen

sarakepermutaatioiden avulla. Tämä on mahdollista vain jos H on täysiasteinen [10].

ALGORITMI TIIVISTETTYNÄ

Esilaskenta: *Syöte:* Kääntyvä parillisuusmatriisi H. *Tulos:* Ekvivalentti parillisuusmatriisi muodossa $\left(\begin{smallmatrix} A & B & T \\ C & D & E \end{smallmatrix}\right)$ niin, että $-ET^{-1}B + D$ on kääntyvä.

1. H:n saattaminen likimääräiseen alakolmiomuotoon

$$H = \begin{pmatrix} A & B & T \\ C & D & E \end{pmatrix}$$

rivi- ja sarakepermutaatioiden avulla, pyrkien minimoimaan väli g.

2. Tulon

$$\begin{pmatrix} I & 0 \\ -ET^{-1} & I \end{pmatrix} \begin{pmatrix} A & B & T \\ C & D & E \end{pmatrix} = \begin{pmatrix} A & B & T \\ -ET^{-1}A + C & -ET^{-1}B + D & 0 \end{pmatrix} \tag{2.1}$$

suorittaminen Gauss -Jordan menetelmällä, jotta $-ET^{-1}B+D$ voidaan todeta kääntyväksi. Jos tarpeellista, suoritetaan sarakepermutaatioita, jotta kääntyvyys toteutuu.

Koodaus: *Syöte:* Parillisuusmatriisi muotoa

$$\begin{pmatrix} A & B & T \\ C & D & E \end{pmatrix}$$

niin, että $-ET^{-1}B + D$ on kääntyvä, ja vektori $\boldsymbol{s} \in \mathbb{F}^{n-m}$.

Tulos: Vektori $\boldsymbol{x} = (\boldsymbol{s}, \boldsymbol{p_1}, \boldsymbol{p_2})$, $p_1 \in \mathbb{F}^g$, $p_2 \in \mathbb{F}^{m-g}$ niin, että $H\boldsymbol{x}^T = \boldsymbol{0}^T$.

1. Ratkaise $\boldsymbol{p_1}$ yhtälöstä $\boldsymbol{p_1}^T = -\phi^{-1}(-ET^{-1}A + C)\boldsymbol{s}^T$.

2. Ratkaise $\boldsymbol{p_2}$ yhtälöstä $\boldsymbol{p_2}^T = -T^{-1}(A\boldsymbol{s}^T + B\boldsymbol{p_1}^T)$.

2.2 Dekoodaus

LDPC-koodien purkamisessa käytettävien algoritmien perhettä kutsutaan Message Passing Algoritmeiksi (MPA) [6]. Gallager esitteli LDPC-koodien yhteydessä erään tällaisen dekoodausalgoritmin, joka on tyypillisesti lähes optimaalinen [1][3]. Myöhemmin muut tutkijat ovat myös itsenäisesti löytäneet tämän ja muita samankaltaisia algoritmeja [6]. Message Passing Algoritmeihin lukeutuvia algoritmeja ovat muun muassa Sum-product- (SPA), Belief Propagation- (BPA) ja Bit flipping-algoritmit [3][8]. Message Passing Algoritmit toimivat lähettämällä viestejä muuttujasolmuilta tarkastussolmuille ja takaisin iteratiivisesti pitkin Tanner- tai tekijägraafia, mistä ne ovat myös saaneet nimensä [8][6]. Yleisemmin kuvailtuna MPA:t pyrkivät hyväksikäyttämään sitä, miten monimutkaiset usean muuttujan funktiot jakautuvat yksinkertaisempiin lokaaleihin funktioihin, ja marginalisoimaan, eli laskemaan tekijöihin jaetun muodon kautta, miten kaikkien muiden muuttujien mahdolliset arvot vaikuttavat yhden tietyn muuttujan arvoihin [4]. Bit-flipping-algoritmin tapauksessa tämä tarkoittaa, että marginalisoitavan bitin arvo käännetään sen perusteella, kumpi arvo toteuttaa parillisuusyhtälöt joihin bitti osallistuu. BPA ja muut vähemmän naiivit algoritmit puolestaan laskevat todennäköisyyksiä bittien arvoista, jonka perusteella päätökset tehdään.

Esimerkki 2.1 Olkoon C koodi ja kuvan 2.1 matriisi H sen parillisuusmatriisi $C(H)$. Koodisanan $\boldsymbol{x}$ jäsenyyden koodissa C määrittää yhtälö $H\boldsymbol{x}^T = \mathbf{0}$, jolloin C:n karakteristinen funktio on muotoa

$$\delta(\boldsymbol{x}) = \begin{cases} 1, & \text{jos } H\boldsymbol{x}^T = \mathbf{0} \\ 0 & \text{muulloin.} \end{cases}$$

Tällöin karakteristinen funktio voidaan jakaa tekijöihin seuraavasti.

$$\delta(\boldsymbol{x}) = [x_2 \oplus x_4 \oplus x_6 = 0][x_2 \oplus x_3 \oplus x_5 = 0][x_1 \oplus x_5 \oplus x_6 = 0][x_1 \oplus x_3 \oplus x_4 = 0]$$

Tästä nähdään, että vaikka Tanner-graafin käsitetään yleisesti esittävän itse koodia, esittää se tarkalleen ottaen koodin karakteristisen funktion tekijöihin jaettua muotoa [4].

2.3 Bit-flipping-algoritmi

Bit-flipping-algoritmi ottaa syötteenään suoraan kanavalta vastaanotetut bitit, ja välittää viestejä pitkin Tanner-graafin kaaria myös yksinomaan bitteinä. Muuttujasolmut lähettävät niiden syötteenä saamansa arvot tarkastussolmuille, ja tarkastussolmut lähettävät muuttujasolmuille puolestaan takaisin viestejä, jotka ilmaisevat bitin arvon, joka toteuttaa tarkastussolmun edustaman parillisuusyhtälön. Jos suurin osa muuttujasolmun saamista viesteistä eroaa sen nykyisestä arvosta, kääntää se arvonsa ykkösestä nollaksi, tai toisinpäin [8].

Bit-flipping-algoritmi perustuu siihen, että bitti joka osallistuu suureen määrään toteutumattomia parillisuusyhtälöitä on todennäköisesti itse väärässä. LDPC-koodien harva parillisuusmatriisi levittää bitit niin, ettei sama bitti esiinny useassa parillisuusyhtälössä, mikä auttaa algoritmia konvergoitumaan [8].

Esimerkki 2.2 [8] Olkoon H kuvan 2.1 LDPC-koodin parillisuusmatriisi, $\boldsymbol{x} = (0, 1, 1, 1, 0, 0)$ jonkin kanavan ylitse lähetetty koodisana ja $\boldsymbol{y} = (0, 1, 1, 1, 0, 1)$ vastaanotettu sana. Merkataan muuttujasolmulta tarkastussolmulle lähtevää viestiä M_i:llä, jossa i vastaa vastaanotetun sanan $\boldsymbol{y}$ i:nnettä komponenttia kuin myös i:nnettä muuttujasolmua. Merkataan lisäksi j:nneltä tarkastussolmulta i:nnelle viestisolmulle lähtevää viestiä $E_{j,i}$.

Aluksi muuttujasolmut alustavat viestinsä kanavalta vastaanotetuilla arvoilla, asettamalla $M_i = y_i$, jonka jälkeen muuttujasolmut lähettävät viestinsä tarkastussolmuille, joihin ne ovat yhdistyneet Tanner-graafin kaaria pitkin. c_1 vastaanottaa siis viestit $M_2 = 1$, $M_4 = 1$ ja $M_6 = 1$. Muut tarkastussolmut vastaanottavat samankaltaisesti viestit niihin yhdistyneiltä muuttujasolmuilta. Tämän jälkeen tarkastussolmut lähettävät viestin takaisin jokaiselle niihin yhdistetylle muuttujasolmulle. Nämä viestit sisältävät summan kaikista tarkastussolmun aiemmin saamista viesteistä, paitsi siltä muuttujasolmulta vastaanotetun viestin, jolle kyseinen viesti on nyt menossa. Tällöin c_1 lähettää viestit $E_{1,2} = M_4 \oplus M_6 = 1 \oplus 1 = 0$, $E_{1,4} = M_2 \oplus M_6 = 1 \oplus 1 = 0$ ja $E_{1,6} = M_2 \oplus M_4 = 1 \oplus 1 = 0$ takaisin muuttujasolmuille. Loput tarkastussolmujen lähettämät viestit ovat:

$$E_{2,2} = 1, E_{2,3} = 1, E_{2,5} = 0$$

$$E_{3,1} = 1, E_{3,5} = 1, E_{3,6} = 0$$

$$E_{4,1} = 0, E_{4,3} = 1, E_{4,4} = 1$$

Tämän jälkeen muuttujasolmut tekevät päätöksen bitin kääntemisestä. Päätös bitin kääntämisen puolesta tehdään, jos suurin osa muuttujasolmun vastaanottamista viesteistä eroaa sen nykyisestä arvosta. v_1 vastaanottaa viestit $E_{3,1} = 1$ ja $E_{4,1} = 0$, joten ensimmäisen bitin arvoa ei käännetä. Sama pätee kaikille paitsi kuudennelle bitille, jonka arvo on 1. Koska v_6 vastaanottaa viestit $E_{1,6} = 0$ ja $E_{3,6} = 0$, niin kuudennen bitin arvo käännetään. Uudet muuttujasolmuilta viestisolmuille lähetettävät viestit ovat nyt $\boldsymbol{M} = (0, 1, 1, 1, 0, 0)$.

Lopuksi viestisolmut summaavat yhteen vielä kerran kaikki niille saapuneet viestit ja tarkistavat parillisuusyhtälöiden toteutumisen eli, että $\delta(\boldsymbol{M}) = 1$. Laskettavat parillisuusyhtälöt ovat siis seuraavat.

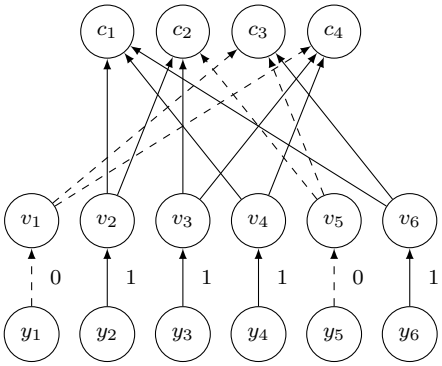
$$M_2 \oplus M_4 \oplus M_6 = 1 \oplus 1 \oplus 0 = 0$$

$$M_2 \oplus M_3 \oplus M_5 = 1 \oplus 1 \oplus 0 = 0$$

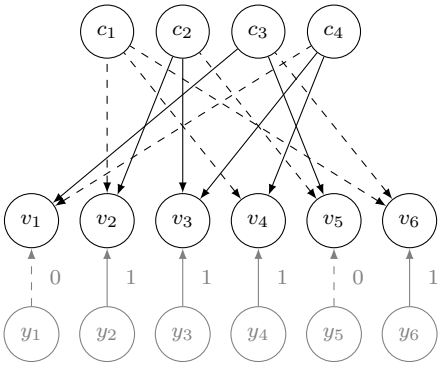
$$M_1 \oplus M_5 \oplus M_6 = 0 \oplus 0 \oplus 0 = 0$$

$$M_1 \oplus M_3 \oplus M_4 = 0 \oplus 1 \oplus 1 = 0$$

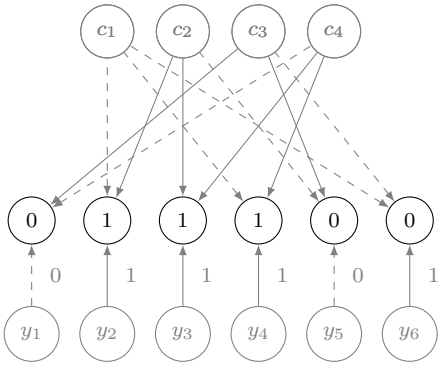
Jokainen neljästä parillisuusyhtälöstä toteutuu, joten algoritmi pysähtyy, ja palauttaa sanan $M = (0, 1, 1, 1, 0, 0)$. Kanavalta vastaanotettu sana on täten purettu ja korjattu [8]. Alla olevat kuvat 2.3-2.6 havainnollistavat algoritmin eri vaiheet. Ykkösen sisältävät viestit on esitetty yhtenäisellä viivalla ja nollan sisältävät viestit katkoviivalla.



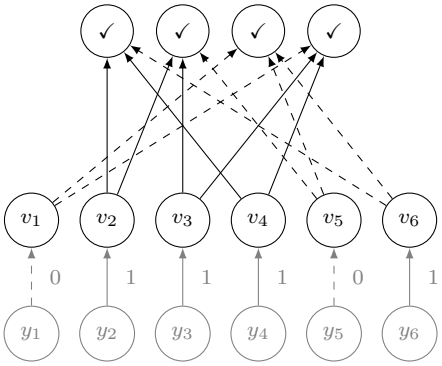
Kuva 2.3: Alustus ja viestit M_i muuttujasolmuilta tarkastussolmuille.



Kuva 2.4: Viestit $E_{j,i}$ tarkastussolmuilta muuttujasolmuille.



Kuva 2.5: Muuttujasolmujen päivitys.



Kuva 2.6: Parillisuusyhtälöiden tarkistus.

2.4 Sum-product-algoritmi

Sum-product-algoritmi on soft decision message passing-algoritmi, joka bittien sijaan välittää viesteinä todennäköisyyksiä bittien arvoista. Algoritmin syötteenä saamien bittien todennäköisyyksiä kutsutaan a priori todennäköisyyksiksi, ja algoritmin palauttamien bittien todennäköisyyksiä a posteriori todennäköisyyksiksi. Algoritmin tavoitteena on laskea maximum a posteriori todennäköisyys (MAP) jokaiselle bitille ehdolla, että kaikki parillisuusyhtälöt toteutuvat, eli $P(x_i = 1|\delta(\boldsymbol{x}) = 1)$ [8].

Nimestään huolimatta, sum-product algoritmin numeerisen vakauden ja implementaation yksinkertaistamisen vuoksi, todennäköisyyksien tuloista pyritään hankkiutumaan kuitenkin eroon käyttämällä log-likelihood-suhdetta (LLR), jolloin kertolaskut voidaan korvata summauksilla [8][3]. Binääriselle satunnaismuuttujalle x log-likelihood suhde määritellään seuraavasti.

$$L(x) = \ln\left(\frac{P(x=0)}{P(x=1)}\right)$$

Jos $P(x = 0) > P(x = 1)$ on $L(x)$ positiivinen, ja päinvastoin. $|L(x)|$ puolestaan on sitä suurempi, mitä luotettavampi x:n arvo on [8].

Bit flipping algoritmin tavoin, muuttujasolmut keräävät ulkoista tietoa niitä naapuroivien tarkastussolmujen kautta. Tämä tieto lähetetään viesteinä $E_{j,i}$ tarkastastussolmulta j muuttujasolmulle i, jossa $E_{j,i}$ on LLR todennäköisyydestä $P_{j,i}^{\text{ext}}$, että bitti i toteuttaa parillisuusyhtälön j. Todennäköisyys, että parillisuusyhtälö j toteutuu, kun bitti i on 1 on

$$P_{j,i}^{\text{ext}} = \frac{1}{2} - \frac{1}{2}\prod_{i\sim}(1 - 2P_i^{\text{int}}),$$

jossa P_i^{int} on tämänhetkinen arvio todennäköisyydestä, että bitti i on 1. Tällöin todennäköisyys, että bitti i on 0 on $1 - P_{j,i}^{\text{ext}}$, joten $E_{j,i}$ saa muodon

$$E_{j,i} = \mathrm{LLR}(P_{j,i}^{\mathrm{ext}}) = \ln\left(\frac{1 - P_{j,i}^{\mathrm{ext}}}{P_{j,i}^{\mathrm{ext}}}\right) = \ln\left(\frac{\frac{1}{2} + \frac{1}{2}\prod_{i\sim}(1 - 2P_i^{\mathrm{int}})}{\frac{1}{2} - \frac{1}{2}\prod_{i\sim}(1 - 2P_i^{\mathrm{int}})}\right).$$

Tässä ilmaisu $\prod_{i\sim}$ tarkoittaa, että tulo otetaan kaikkien niiden muuttujasolmujen yli, jotka ovat kaarella yhdistettynä j:nteen tarkastussolmuun, lukuunottamatta i:nnettä muuttujasolmua.

Hyperbolisen tangentin identiteettejä $\tanh\left(\frac{1}{2}\ln\frac{1-p}{p}\right) = 1 - 2p$ ja $2\tanh^{-1}(p) = \ln\left(\frac{1+p}{1-p}\right)$ käyttämällä yllä oleva yhtälö voidaan edelleen kirjoittaa muotoon

$$E_{j,i} = 2\tanh^{-1}\left(\prod_{i\sim}\tanh\left(\frac{M_{j,i}}{2}\right)\right),$$

jossa $M_{j,i}$ on muuttujasolmun i lähettämä viesti tarkastussolmulle j, ja määritellään seuraavasti [8][3].

$$M_{j,i} = \mathrm{LLR}(P_{j,i}^{\mathrm{int}}) = ln\left(\frac{1 - P_{j,i}^{\mathrm{int}}}{P_{j,i}^{\mathrm{int}}}\right)$$

Kokonais-LLR kullekin muuttujasolmulle i on summa sen syötteenä saamastaan a priori LLR:stä ja kaikista siihen yhdistetyiltä tarkastussolmuilta saaduista LLR:stä

$$L_i = \mathrm{LLR}(P_i^{\mathrm{int}}) = r_i + \sum_{j\in N(v_i)} E_{j,i}.$$

Viestit muuttujasolmuilta tarkastussolmuille eivät kuitenkaan sisällä koko bittikohtaista LLR:ää, vaan ainoastaan sen osan, joka on peräisin kaikilta muilta tarkastussolmuilta. Näin vältetään lähettämästä tarkastussolmuille tietoa, joka niillä jo on. Viesti $M_{j,i}$ muuttujasolmulta i tarkastussolmulle j on siis yllä oleva summa ilman viestiä $E_{j,i}$ [8][3].

$$M_{j,i} = r_i + \sum_{j\sim} E_{j,i}$$

Viestit $E_{j,i}$ pitävät sisällään *tanh*-, sen käänteisfunktion, ja tuloja, jotka tekevät algoritmin implementaatiosta hankalaa. $E_{j,i}$:n yksinkertaistamiseksi $M_{j,i}$ voidaan

jakaa sen merkki- ja itseisarvo-osiin

$$M_{j,i} = \alpha_{j,i}\beta_{j,i} = \text{sign}(M_{j,i})|M_{j,i}|,$$

jonka jälkeen $E_{j,i}$ voidaan manipuloida muotoon

$$E_{j,i} = (\prod_{i\sim} \alpha_{j,i})\phi(\sum_{i\sim} \phi(\beta_{j,i})),$$

missä

$$\phi(x) = -\ln\tanh\left(\frac{x}{2}\right) = \ln\frac{e^x+1}{e^x-1}.$$

$E_{j,i}$:n muotoa voidaan kuitenkin yksinkertaistaa edelleen, sillä $\beta_{j,i}$:n pienin arvo dominoi summan $\sum_{i\sim}\phi(\beta_{j,i})$ arvoa, jolloin summaa voidaan approksimoida minimillä. Lisäksi koska $\phi(\phi(x)) = x$, saa $E_{j,i}$ muodon

$$E_{j,i} \approx (\prod_{i\sim} \alpha_{j,i}) \min_{i\sim} \beta_{j,i}.$$

Molemmissa tapauksissa merkkien tulo voidaan laskea modulo 2 summalla, jolloin jälkimmäinen muoto vaatii ainoastaan vertailu- ja yhteenlaskuoperaatioita. Tätä muotoa kutsutaan myös Min-Sum-algoritmiksi [1][8][3].

ALGORITMI TIIVISTETTYNÄ

Syöte: Vektori $\boldsymbol{r}$, joka sisältää bittikohtaiset a priori todennäköisyydet, ja iteraatioiden maksimimäärä.

Tulos: Purettu koodisana $\boldsymbol{z}$ tai ei mitään, jos maksimi-iteraatioiden määrä saavutettiin ennen sanan löytymistä.

1. Alusta muuttujasolmujen viestit $M_{j,i}$ bittikohtaisilla a priori todennäköisyyksillä r_i.

2. Jokaista tarkastussolmua kohden laske $E_{j,i}$.

3. Jokaista muuttujasolmua kohden laske kokonais LLR L_i.

4. Rakenna vektori $\boldsymbol{z}$ asettamalla $z_i = \begin{cases} 1, & \text{jos } L_i \leq 0 \\ 0, & \text{jos } L_i > 0. \end{cases}$

5. Jos $\delta(\boldsymbol{z}) = 1$ palauta $\boldsymbol{z}$. Jos taas maksimi iteraatioiden määrä on saavutettu pysäytä algoritmin suoritus. Muuten, jokaista muuttujasolmua kohden päivitä viestien $M_{j,i}$ arvot asettamalla $M_{j,i} = r_i + \sum_{j\sim} E_{j,i}$ ja palaa kohtaan 2.

3 LDPC-koodit 5G- ja 6G-verkoissa

5G-verkkojen käyttöönotto ympäri maailmaa on joko laajalti meneillään tai jo toteunut ja tutkimusta B5G- (Beyond 5G) ja 6G-verkkojen teknologioita varten on tehty jo jonkin aikaa. Tämän kaltaisiin verkkoihin osallistuvien mobiililaitteiden määrän oletetaan joidenkin lähteiden mukaan olevan 17 miljardia vuonna 2030, ja dataliikenteen arvioidaan olevan yli 5000 eksatavua kuussa. Tämän lisäksi M2M-yhteyksien (Machine to Machine) määrän oletetaan kasvavan eksponentiaalisesti. Viidennen sukupolven mobiiliverkkojen oletetaan saavuttavan kapasiteettinsa myös samana vuonna, jolloin tarvitaan jälleen uuden sukupolven teknologioita vastaamaan yhä kasvaviin vaatimuksiin. Jokaisella siirtymällä sukupolvesta seuraavaan kasvu suoritustehossa on ollut 10-100-kertainen. ITU:n (International Telecommunication Union) määrittelemien KPI-tavoitteiden (Key Performance Indicator) mukaan siirtymä kuudennen sukupolven verkkoihin edellyttäisi muun muassa parannuksia datan siirtonopeuteen, koettuun siirtonopeuteen, latenssiin, paikannukseen, luotettavuuteen, samanaikaisten yhteyksien tiheyteen sekä spektri- ja energiatehokkuuteen. Datan huippusiirtonopeuden kohdalla vaatimukset ovat nostaa nykyistä rajaa 20Gbit/s:stä jopa 1Tbit/s:iin. Tämän lisäksi 6G:n on tuettava useita eri käyttökohteita, jotka vaihtelevat teollisuudesta aina henkilökohtaisiin mobiililaitteisiin asti. Tämä tarkoittaa myös sitä, että 6G:n kanavakoodausmenetelmien määrittäminen vaatii yhä useampien tavoitteiden optimointia sekä kompromissien punnitsemista sopivien ratkaisujen löytämiseksi [11][12][13][14].

2G:stä lähtien lähes joka sukupolvi on ottanut käyttöön uuden kanavakoodausmenetelmän. 2G:ssä käytettiin konvoluutiokoodeja, 3G- ja 4G-verkoissa Turbo-koodeja ja niiden muunnelmia sekä 5G:ssä Polar- ja LDPC-koodeja. Joka sukupolven verkko on kuitenkin perustunut paljolti edellisen arkkitehtuuriin, ja näin oletetaan myös tapahtuvan 6G:n kohdalla. Vielä ei ole kuitenkaan selvää kuinka paljon 5G-teknologioita voidaan käyttää uudelleen, kuinka monet vaativat päivityksiä ja kuinka paljon on rakennettava alusta asti [14][11].

Tässä luvussa kartoitetaan tilannetta virheenkorjausmenetelmien kannalta, keskittyen etenkin LDPC-koodien ja muiden vallitsevien menetelmien potentiaaliin kanavakoodausmenetelmänä 6G:ssä, ja näin ollen vastataan tutkimuskysymyksiin **TK1** ja **TK2**.

3.1 LDPC-koodit 5G-verkoissa

5G-verkkojen käyttökohteet on jaettu kolmeen kategoriaan, jotka ovat eMBB (Enhanced Mobile Broadband), URLLC (Ultra-Reliable Low-Latency Communications) ja mMTC (Massive Machine Type Communications). Karkeasti jaoteltuna eMBB keskittyy nopeaan dataliikenteeseen ja korkeaan datan määrään, URLLC matalaan viiveeseen ja mMTC yhteyksien suureen määrään [15]. Luvussa 2 mainittiin joitakin pitkien LDPC-koodien eduista. Koska eMMB keskittyy suuren datan määrän lähettämiseen nopeasti ja tehokkaasti, esimerkiksi multimediasuoratoiston muodossa, on LDPC-koodien hyöty helposti havaittavissa tälle käyttökohteelle. 3GPP (3rd Generation Partnership Project) päättikin valita LDPC-koodit eMBB:n dataliikenteen kanavakoodeiksi. Vähemmän dataa vaativalle ohjauskanavalle sen sijaan (alle 400 bittiä lohkoa kohden) valittiin Polar-koodit [15]. 3GPP:n vuoden 2026 huhtikuussa julkaisema multipleksauksen ja kanavakoodauksen spesifikaatio kuitenkin määrittää LDPC-koodit kaikkien TrCH (Transport Channel) kanavien koodausmenetelmäksi, paitsi BCH:n (Broadcast Channel), joka käyttää Polar-koodeja. Tämä tarkoittaa

sitä, että muutkin käyttökohteet eMMB:n lisäksi hyödyntävät LDPC-koodeja datalähetyksissään tutkielman kirjoitushetkellä. Polar-koodeja käytetään lisäksi kaikilla ohjauskanavilla [16].

5G:lle standardoidut LDPC-koodit ovat kvasi-syklisiä, ja ne voidaan rakentaa kahdesta eri pohjamatriisista. Näiden pohjamatriisien tukevat informaatiobittimäärät vaihtelevat välillä 40-8443. Informaation suhteellinen määrä (Code rate) taas on pienimmillään 1/5 ja suurimmillaan jopa 0.95. Lopullinen parillisuusmatriisi saavutetaan laajentamalla jokaista pohjamatriisin arvoa $Z \times Z$ siirtomatriiseilla, joita nimensä mukaisesti siirretään syklisesti oikealle ennalta määritettyjen siirtovakioiden osoittamasti. Kummallekin pohjamatriisille on määritelty 51 eri kokoista siirtomatriisia, jotka mahdollistavat koodin optimoinnin eri käyttötarkoituksia varten. Korkeammat siirtoarvot Z mahdollistavat enemmän rinnakkaislaskentaa koodien purkamisessa. Kuitenkin tämän konstruktion päätavoitteena on toiminta lähellä kanavan kapasiteettia hyvällä virhetasolla (Error Floor) [17][18].

3.2 Menetelmien vertailu

Kirjallisuudessa tyypillinen tapa määrittää langattoman kommunikaatiojärjestelmän suorituskykyä on mitata lähetyksessä havaitun BER:n tai BLER:n (Block Error Rate) suuruutta SNR:ää (Signal to Noise Ratio) vasten. Toisin sanoen, piirtää kuvaaja BER:stä tai BLER:stä SNR:n funktiona. Aiemmin luvussa 3.1 mainittu virhetaso, joka tarkoittaa tämän funktion käyrän loivenemista, on myös yleisesti menetelmien välillä vertailtu ominaisuus. Mittauksissa ja simulaatioissa käytettyihin lisäparametreihin lukeutuvat: informaation määrä (Code rate), signaalin modulaatiomenetelmä, dekoodausalgoritmit, dekoodausalgoritmin maksimi-iteraatioiden määrä ja dekoodausalgoritmien kompleksisuus.

Tässä osasassa kootaan ja tarkastellaan kirjallisuudessa raportoituja tuloksia, keskittyen kolmeen nykytutkimuksessa yleisimmin käsiteltyyn virheenkorjausmenetelmään: LDPC-, Polar- ja Turbo-koodeihin.

Taulukko 3.1: Koottuja BER/BLER tuloksia virheenkorjausmenetelmien vertailuista kirjallisuudessa. (Informaatiolohkon pituus merkattu K:lla, koko lohkon- N:llä.)

Artikkeli	Vertailussa	Kanava ja modulaatiomenetelmä	Lohkonpituus	Määrä	Dekoodaus	Tulokset
[19]	LDPC, Polar	AWGN, BPSK	K=64–1024	1/3	LDPC: 5G standard layered min-sum, (15 iter.); Polar: CRC-SCL ($L = 8$, CRC16)	Polar-koodit suoriutuvat paremmin lyhyillä pituuksilla. Suorituskyky on samankaltaista keskipituuksilla
[20]	LDPC, Turbo, BCH	AWGN, BPSK	N=1024	1/2, 1/3, 4/7	LDPC: BP; Turbo: MAP; BCH: matriisitulo	BCH suoriutuu parhaiten, mutta kykenee korjaamaan vain 2 bittiä; LDPC suoriutuu hyvin kaikilla pituuksilla; Turbo parhaiten lyhyillä pituuksilla.
[21]	LDPC, Turbo	PLC noise + AWGN, BPSK, impulse noise	Turbo: N=6480; LDPC: N=5184, 6480, 8640	Turbo: 1/2; LDPC: 5/6, 2/3, 1/2	LDPC: SPA; Turbo: MAP	QC-LDPC-koodit suoriutuvat parhaiten lohkopituudesta ja määrästä riippumatta kaikilla paitsi kaikkein matalimmilla SNR arvoilla, jolloin Turbo koodit suoriutuvat marginaalisesti paremmin. Lohkonpituus 8640 ja määrä 1/2 saavutti parhaimmat tulokset.
[14]	LDPC, Polar, Turbo	AWGN, BPSK	N=128	1/2	LDPC: SPA, Min-sum (20 iter); Turbo: Log-MAP; Polar: SC ($L = 1$) ja SCL ($L = 4, 8, 16, 32$)	Turbo-koodit pärjäävät parhaiten pienellä SNR arvolla, jonka jälkeen Polar- ja LDPC-koodit suoriutuvat paremmin. SNR:n kasvaessa edelleen LDPC-koodit suoriutuvat Polar-koodeja paremmin etenkin SPA dekoodauksella.
[15]	LDPC, Polar	AWGN, Rician, Raleygh, BPSK, QPSK, 16-QAM, 256-QAM	N=64–13056	1/3, 1/2, 2/3, 3/4, 4/5	LDPC: Min-sum (16 iter.), offset Min-sum (16 iter.); Polar: D-CRC SCL ($L = 16$)	Polar-koodit suoriutuivat paremmin lyhyemmillä koodin pituuksilla, ja LDPC koodit taas paremmin pidemmillä. Lyhyemmät 1024 mittaiset LDPC-koodit suoriutuivat paremmin kuin saman mittaiset Polar-koodit Offset Min-sum dekoodauksella. Molemmat koodit suoriutuivat parhaiten AWGN kanavalla.
[17]	LDPC, Polar, Turbo, TBCC	AWGN, QPSK, 16-QAM, 64-QAM	K=40–6144 (Polar enintään 1706)	1/5, 1/3, 1/2, 5/6	LDPC: normalized min-sum, layered scheduling (20 iter.); Polar: SC ($L = 1$) ja CRC-SCL ($L = 8, 16$); Turbo: Max-Log-MAP (8 iter.); TBCC: Viterbi	Polar-koodit suoriutuvat parhaiten lyhyillä lohkopituuksilla ja tarjoavat noin 1–3 dB paremman suorituskyvyn kuin LDPC- ja Turbo-koodit. LDPC-koodien suorituskyky verrattuna Turbo-koodeihin on parempi suurilla lohkopituuksilla, ja päinvastoin.
[22]	LDPC, Polar	AWGN, Rician, Raleygh, Nakagami BPSK, QPSK, 16-QAM, 256-QAM	-	1/2, 3/4	LDPC: Min-sum; Polar: SC	LDPC-koodit suoriutuvat hyvin suurilla lohkopituuksilla AWGN- ja Nakagami kanavilla. Polar-koodit suoriutuivat parhaiten lyhyillä lohkopituuksilla. Suorituskyky oli molemmilla menetelmillä paras AWGN-kanavalla.
[12]	LDPC, Polar, Turbo	AWGN, QPSK	K=100–8000	1/5, 1/3, 1/2, 2/3, 8/9	LDPC: Adjusted Min-sum (25 iter.); Polar: SCL ($L = 32$); Turbo: Max-Log-MAP (8 iter.)	Polar-koodit suoriutuvat parhaiten lyhyillä lohkoilla ($K \leq 400$) koodin määrästä riippumatta. Pienillä määrillä (1/5) LDPC-koodit suoriutuivat paremmin kuin Polar-koodit kun $K = 1000$. Suuremmilla lohkoilla LDPC-koodit suoriutuivat parhaiten määrästä riippumatta. Turbo-koodit eivät osoittautuneet edulliseksi kumpaankaan menetelmään verrattuna millään parametreilla.
[23]	LDPC, Polar, Turbo, Convolutional	AWGN, QPSK	K=256–8192	1/3, 1/2, 2/3, 5/6	LDPC: Layered Min-sum (16 iter.); Polar: SCL ($L = 8$); Turbo ja Convolutional: Max-Log-MAP (8 iter.)	Convolutional-koodit suoriutuivat kaikilla mittareilla huonoiten. Muiden menetelmien suorituskyky oli erittäin lähellä toisiaan, etenkin suurilla lohkonpituuksilla. Tyypillisesti Turbo-koodit suoriutuivat parhaiten kaikilla määrillä paitsi 5/6, jolloin LDPC-koodit osoittivat marginaalisesti parempaa suorituskykyä.

Taulukossa 3.1 on koottuna kirjallisuudessa raportoituja tuloksia eri virheenkorjausmenetelmille toteutetuista simulaatioista ja vertailuista. Näiden tulosten nojalla voidaan tehdä seuraavat johtopäätökset:

1. Polar-koodit suoriutuvat LDPC- ja Turbo-koodeja paremmin lyhyillä lohkonpituuksilla. Polar-koodit saavuttivat lyhyillä lohkopituuksilla muita menetelmiä paremman suorituskyvyn kaikissa tarkastelluissa vertailuissa riippumatta koodin määrästä ja dekoodausalgoritmin yksityiskohdista, lukuun ottamatta B. Tahir et al. ja A. Gautam et al. artikkeleissa raportoituja tuloksia [23][14]. B. Tahir et al. artikkelin tapauksessa tämä johtunee siitä, että käytetyn dekoosausalgoritmin SCL (Successive Cancellation List) peruutuslistan koko oli ainoastaan 8. Algoritmin suorituskyky kasvaa BER:n suhteen peruutuslistaa kasvattamalla, ja artikkelissa mainitaan, että niinkin hyvä tulos oli yllättävää vaikkei käytetty menetelmä ollut optimaalisin. A. Gautam et al. artikkelin poikkeavia tuloksia ei kuitenkaan voida selittää samalla argumentillä, sillä korkeammat listan koot, aina 32:n asti olivat vertailussa mukana. Menetelmien BER-vertailu ei ole artikkelin keskipisteenä, ja tähän liittyvä sisältö ei tarjoa selitystä ilmiölle, joten tulosten poikkeavuuden syy jää epäselväksi.

2. LDPC-koodit suoriutuvat Polar- ja Turbo-koodeja paremmin pitkillä lohkonpituuksilla. Sae-Young Chung et al. julkaisussa ehdotetaan, että optimaalisten LDPC-koodien suorituskyky lähenee kanavan kapasiteettia lohkonpituuden kasvaessa [9]. Ottaen tämän huomioon, ei ole yllättävää, että LDPC-koodit suoriutuvat muihin menetelmiin nähden myös parhaiten pitkillä lohkonpituuksilla. Huomattakoon, että B. Tahir et al. tutkimuksessa ero Turbo-koodeihin on häilyvä [23]. Muissa tutkimuksissa havaitut suuremmat erot voidaan selittää käytetyllä Min-sum algoritmin variaatiolla. B. Tahir et al. käyttivät vertailussa Layered Min-sum-algoritmia, joka optimoi BER:n sijasta rinnakkaislaskentaa. Muissa artikkeleis-

sa esiintyneet Adjusted Min-sum ja Offset Min-sum algoritmit puolestaan pyrkivät lieventämään Min-sum-algoritmissä käytettyjen approksimaatioiden lisäämää epätarkkuutta, joka kasvattaa näiden algoritmien BER-suorituskykyä huomattavasti lähemmäs alkuperäistä Sum-Product algoritmia [15][12].

Esitetyt johtopäätökset ovat yhdenmukaisia myös sen kanssa, miten koodausmenetelmien käyttökohteet ovat jakautuneet 3GPP:n 5G standardissa. Kuten osassa 3.1 aiemmin mainittiin, Polar-koodit ovat käytössä ohjauskanavilla ja LDPC-koodit datakanavilla. Kirjallisuudesta kootut tulokset selvästi siis tukevat 3GPP:n standardissa määriteltyä työnjakoa kanavakoodausmenetelmille.

Vaikka Turbo-koodit eivät suoriutuneet vertailussa kauttaaltaan hyvin LDPC- ja Polar-koodeihin nähden, oli niillä kuitenkin A. Gautam et al. ja G. Prasad et al. suorittamissa vertailuissa etu matalilla SNR arvoilla. Tämän perusteella voidaan arvioida, että Turbo-koodit olisivat varteenottava virheenkorjausmenetelmä tilanteissa, joissa signaalin laatu on huono.

3.3 Näkymät 6G:tä kohti

Seuraavan sukupolven verkkojen käyttökohteet eivät ole vielä kauttaaltaan lukkoon lyötyjä. Kuitenkin useat eri yritykset ja organisaatiot ovat julkaisseet raportteja, jotka pyrkivät niitä määrittämään. Yksi tällainen raportti [24] koostaa nämä käyttökohteet lisäämällä 5G:lle määritettyjen käyttökohteiden eMBB, URLLC ja mMTC joukkoon hybridikäyttökohteet URLLC-MBB, mMTC-MBB ja URLLC-mMTC. Uudet käyttökohteet pyrkivät lisäämään tukea muun muassa autonomisille ajoneuvoille, älykaupungeille, lisätty- (Augmented Reality) sekä laajennettu todellisuus (Extended Reality) sovelluksille, ja tekoäly- sekä koneoppimissovelluksille. Esimerkiksi URLLC-mMTC pyrkii yhdistämään URLLC KPI-elementtejä, kuten korkean luotettavuuden ja matalan latenssin mMTC:n mahdollistaakseen tuen autonomiselle

liikenteelle [12].

Käyttökohteet ja sovellukset, joita kanavakoodausmenetelmien täytyy tukea ovat siis selkeästi kasvamassa edellisen sukupolven verkkoihin nähden. KPI, joka selkeiten liittää kanavakoodausmenetelmät eri käyttökohteisiin, on luotettavuus. Tätä myös BER/BLER vastaan SNR-mittaukset pyrkivät määrittämään, ja suurin etu näissä mittauksissa on, että tulokset ovat toistettavissa ja yksiselitteisesti verrattavissa keiden tahansa osapuolien kesken. Tämä ei ole yhtä itsestäänselvää muiden KPI mittareiden kohdalla, joihin kanavakoodaus liittyy [13]. Luotettavuuden ohella muita KPI-tavoitteisiin liittyviä haasteita ovat:

Monipuolisuus. Kuten jo todettiin, tuettavien käyttökohteiden määrä on kasvamassa jälleen 6G-verkkoihin siirryttäessä. Tämä trendi pystyttiin kuitenkin jo havaitsemaan neljännestä sukupolvesta viidenteen siirryttäessä, kun käyttöön otettiin kaksi eri virheenkorjausmenetelmää yhden sijasta, Polar- ja LDPC-koodit, joista molemmat tukevat vielä lisäksi monia eri konfiguraatioita eri skenaarioita varten.

Dekoodauksen monimutkaisuus ja tehokkuus. Dekoodausalgoritmien implementaation monimutkaisuus ja tehokkuus on eräs suuri ongelma 6G:tä lähestyttäessä. Käytännön puolesta virheenkorjausmenetelmien suurin rajoite ei ole koskaan ollut niiden etäisyys kanavan kapasiteetista, vaan dekoodaajan monimutkaisuus. Kanavakoodauksen kriittisestä roolista johtuen algoritmejä ei ajeta yleiskäyttöisillä prosessoreilla, vaan dedikoiduilla ASIC (Application-Specific Integrated Circuit) piireillä. Näiden piirien tulee ottaa kaiken edellä mainitun lisäksi huomioon pinta-ala- ja energiatehokkuus. Näin ollen ongelma-avaruuden ulottuvuusten määrä kasvaa entisestään. Lisäksi yhtä tavoitetta lähemmäksi pääseminen tarkoittaa monesti toisesta erkaantumista. Esimerkiksi näissä ASIC-piireissä energiatehokkuuden (*Joule/bit*) parantaminen maksetaan spektrin tehokkuudessa $\left(\frac{\text{bit/s}}{\text{Hz}}\right)$ [12][13].

Latenssi. Kanavakoodaus lisää prosessointiaikaa niin lähettäjän kuin vastaanottajan puolella. Tämä on kuitenkin välttämätöntä luotettavan kommunikaatiokanavan saavuttamiseksi. Muutkin tekijät myötävaikuttavat matalaan latenssiin, mutta käytännössä suurin vaikutus tulee uudelleenlähetyksistä. Kerralla onnistunut dekoodaus välttää tarpeen lähetyksen toistamiseen, joten tehokas virheenkorjaus on latenssin kannalta kriittistä. 5G:ssä tätä tavoitetta edistettiin matalammalla koodin määrällä ja modulaatioasteella. Tämä kuitenkin jälleen maksettiin spektrin tehokkuudessa [12][13].

Datansiirron nopeus. Suunnitellut enimmillään THz taajuuksilla toimivat spektriteknologiat mahdollistavat Tbit/s siirtonopeudet. Tämän luokan siirtonopeudet puolestaan mahdollistavat eMMB-palveluita kuten AR ja XR. Neljännestä viidenteen sukupolveen siirryttäessä LDPC-koodit mahdollistivat sen aikaiset harppaukset datansiirrossa, niiden rinnakkaislaskentaa hyödyntävien dekoodausalgoritmien ja suurille datagrammeille edullisten ominaisuuksiensa vuoksi [11][13].

Suuria harppauksia on vielä tehtävä, jotta kaikki KPI-tavoitteet voidaan saavuttaa. M. Chowdhury et al. arvioivat artikkelissaan, että eniten työtä on tehtävä verkon luotettavuuden saralla [11]. A. Gautam et al. puolestaan arvioivat joidenkin URLLC-sovellusten vaativan jopa 10^{-9} BER arvoja, joka korostaa luotettavuuden tarvetta entisestään [14]. Nykyisten mobiiliverkkojen virheenkorjausmenetelmät eivät sellaisenaan riitä takaamaan uuden sukupolven vaatimuksia. Muitakin kuin tässä tutkielmassa käsiteltyjä kanavakoodausmenetelmiä tutkitaan 6G:n käyttökohteita varten. Lisäksi on esitetty yhdistettyjä dekoodausalgoritmeja, jotka purkavat useampaan kuin yhteen koodausmenetelmään perustuvia koodeja, sekä koneoppimista ja tekoälyä hyödyntäviä lähestymistapoja [25][12][11]. Tutkielmassa käsitellyt LDPC-, Polar- ja Turbo-koodit lienevät kuitenkin vahvoja ehdokkaita seuraavan sukupolven koodaustarpeita varten. Tämä heijastuu nykyisen tutkimuksen kiinnostuksessa

etenkin LDPC- ja Polar-koodeja kohtaan [12]. Nämä menetelmät ovat lisäksi olleet jo osana aiempia standardeja, ja niitä on onnistuneesti implementoitu eri käyttötarkoituksia varten, joka lisää luottamusta niitä kohtaan. Odotettavissa on että näitä menetelmiä joudutaan kuitenkin uudistamaan, jotta ne täyttävät niille asetetut vaatimukset [12].

4 Yhteenveto

Tässä tutkielmassa perehdyttiin LDPC-koodeihin teorian, sovellusten ja vertailun näkökulmista. Teoriaa käsittelevässä luvussa 2 käytiin läpi mitä LDPC-koodit ovat, millaisia eroja niillä on klassisiin menetelmiin, miten niitä voidaan rakentaa ja millaisia ominaisuuksia niillä on. Tämän lisäksi käsiteltiin yleinen koodausalgoritmi, joka toimittaa yksinkertaisempiin menetelmiin nähden huomattavasti parempaa suorituskykyä aikakompleksisuuden suhteen. Luvun lopulla perehdyttiin LDPC-koodien dekoodauksessa käytettyihin iteratiivisiin algoritmeihin. Esitellyt Bit-flipping-, Sum-Product- ja Min-Sum-algoritmit ovat osa Message-Passing-algoritmien perhettä ja yhdessä oleellisimmat LDPC-koodien dekoodausmenetelmät. Nämä menetelmät ovat joko sellaisenaan, tai erikoistuneemmissa muodoissa käytössä langattoman verkkoliikenteen teknologioissa, ja ovat aktiivinen tutkimuskohde koodausteorian- ja tietoliikenteen alalla.

Luvun 3 alussa pohjustettiin lyhyesti perustavanlaatuisia haasteita 6G-siirtymään liittyen, jonka jälkeen tutustuttiin LDPC-koodien rooliin nykysukupolven mobiiliverkoissa. Tämän jälkeen aliluvussa 3.2 kootaan kirjallisuudessa esitettyjä BER/BLER-mittausten, ja vertailujen tuloksia vallitsevien menetelmien kesken. Nämä menetelmät ovat LDPC-koodien lisäksi Polar- ja Turbo-koodit. Koottujen tulosten perusteella vastataan tutkimuskysymykseen **TK1: "Miten LDPC-koodien suorituskyky vertautuu muihin vallitseviin menetelmiin?"** LDPC-koodien suurin teoriassa esitetty vahvuus on se, että ne niiden BER suorituskyky lähenee kanavan ka-

pasiteettia, mitä suurempi koodin pituus on. Tämä heijastuu vertailussa esitettyihin tuloksiin, joissa LDPC-koodit suoriutuivat lähes aina parhaiten pisimmillä lohkonpituuksilla. Polar-koodit puolestaan suoriutuivat muita menetelmiä paremmin lyhyillä lohkonpituuksilla. Menetelmien käyttökohteet 5G:ssä on myös jaoteltu niiden edellä mainittujen vahvuuksien mukaan. LDPC-koodien teoreettisia ominaisuuksia on siis selvästi onnistuneesti hyödynnetty käytännössä. Yllättävästi Turbo-koodit suoriutuivat LDPC-koodeja paremmin muutamassa mittauksessa matalilla SNR-arvoilla.

Lopuksi osassa 3.3 perehdyttiin kanavakooduksen kannalta olennaisimpiin 6G:n KPI-tavoitteisiin, ja pyrittiin samalla vastaamaan tutkimuskysymykseen **TK2: "Millaisia haasteita 6G-siirtymä asettaa kanavakoodaukselle, ja riittävätkö nykyiset menetelmät vastaamaan niihin?"** Käyttökohteiden ja ratkaistavien ongelmien määrä on selvästi kasvussa nykysukupolven verkkoihin nähden, ja tavoitteiden vaatimustaso on myös jälleen askeleen korkeammalla. Näiden vaatimusten arvellaan olevan liikaa nykyisille koodausmenetelmille ja niiden implementaatioille. Kuitenkin nykytutkimuksessa ollaan edelleen kiinnostuneita etenkin LDPC- ja Polar-koodeista, josta voidaan arvella, että menetelmillä on potentiaalia vielä 6G-verkkojakin varten.

Lähdeluettelo

[1] R. Gallager, "Low-density parity-check codes", *IRE Transactions on Information Theory*, vol. 8, nro 1, s. 21–28, 1962. DOI: 10.1109/TIT.1962.1057683

[2] R. Tanner, "A recursive approach to low complexity codes", *IEEE Transactions on Information Theory*, vol. 27, nro 5, s. 533–547, 1981. DOI: 10.1109/TIT.1981.1056404

[3] W. E. Ryan, "An introduction to LDPC codes", *CRC Handbook for Coding and Signal Processing for Recording Systems*, vol. 5.2, s. 1–23, 2004.

[4] F. Kschischang, B. Frey ja H.-A. Loeliger, "Factor graphs and the sum-product algorithm", *IEEE Transactions on Information Theory*, vol. 47, nro 2, s. 498–519, 2001. DOI: 10.1109/18.910572

[5] T. R. ja R. Urbanke, *Modern Coding Theory*. Cambridge University Press, 2008.

[6] A. Shokrollahi, "LDPC Codes: an Introduction", teoksessa *Coding, Cryptography and Combinatorics*. Basel: Birkhäuser Basel, tammikuu 2004, s. 85–110.

[7] K. S. Kavitha Sunil Poorna Jayaraj, "Message Passing Algorithm: A Tutorial Review", *IOSR Journal of Computer Engineering*, vol. 2, nro 2278-0661, s. 12–24, 2012. url: https://api.semanticscholar.org/CorpusID:61240752

[8] S. Johnson, "Introducing Low-Density Parity-Check Codes", toukokuu 2010. url: https://www.researchgate.net/publication/228977165_Introducing_Low-Density_Parity-Check_Codes

[9] S.-Y. Chung, G. Forney, T. Richardson ja R. Urbanke, "On the design of low-density parity-check codes within 0.0045 dB of the Shannon limit", *IEEE Communications Letters*, vol. 5, nro 2, s. 58–60, 2001. DOI: 10.1109/4234.905935

[10] T. Richardson ja R. Urbanke, "Efficient encoding of low-density parity-check codes", *IEEE Transactions on Information Theory*, vol. 47, nro 2, s. 638–656, 2001. DOI: 10.1109/18.910579

[11] M. Z. Chowdhury, M. Shahjalal, S. Ahmed ja Y. M. Jang, "6G Wireless Communication Systems: Applications, Requirements, Technologies, Challenges, and Research Directions", *IEEE Open Journal of the Communications Society*, vol. 1, s. 957–975, 2020. DOI: 10.1109/OJCOMS.2020.3010270

[12] M. Rowshan, M. Qiu, Y. Xie, X. Gu ja J. Yuan, "Channel Coding Toward 6G: Technical Overview and Outlook", *IEEE Open Journal of the Communications Society*, vol. 5, s. 2585–2685, 2024. DOI: 10.1109/OJCOMS.2024.3390000

[13] H. Zhang ja W. Tong, "Channel Coding for 6G Extreme Connectivity—Requirements, Capabilities, and Fundamental Tradeoffs", *IEEE BITS the Information Theory Magazine*, vol. 3, nro 1, s. 54–66, 2023. DOI: 10.1109/MBITS.2023.3322978

[14] A. Gautam, P. Thakur ja G. Singh, "Advanced channel coding schemes for B5G/6G networks: State-of-the-art analysis, research challenges and future directions", *International Journal of Communication Systems*, vol. 37, toukokuu 2024. DOI: https://doi.org/10.1002/dac.5855

[15] K. Navin, K. Deepak ja P. Gaurav, "A review of channel coding schemes in the 5G standard.", *Telecommunication Systems*, vol. 83, s. 423–448, kesäkuu 2023. DOI: https://doi.org/10.1007/s11235-023-01028-y

[16] ETSI (3GPP), "5G; NR; Multiplexing and channel coding (3GPP TS 38.212 version 19.3.0 Release 19)", European Telecommunications Standards Institute, Technical Specification ETSI TS 138 212 V19.3.0, huhtikuu 2026. url: https://www.etsi.org/deliver/etsi_ts/138200_138299/138212/19.03.00_60/ts_138212v190300p.pdf

[17] D. Hui, S. Sandberg, Y. Blankenship, M. Andersson ja L. Grosjean, "Channel Coding in 5G New Radio: A Tutorial Overview and Performance Comparison with 4G LTE", *IEEE Vehicular Technology Magazine*, vol. 13, nro 4, s. 60–69, 2018. DOI: 10.1109/MVT.2018.2867640

[18] F. Li et al., "Review on 5G NR LDPC Code: Recommendations for DTTB System", *IEEE Access*, vol. 9, 2021.

[19] S. Belhadj ja M. L. Abdelmounaim, "On error correction performance of LDPC and Polar codes for the 5G Machine Type Communications", teoksessa *2021 IEEE International IOT, Electronics and Mechatronics Conference (IEMTRONICS)*, 2021, s. 1–4. DOI: 10.1109/IEMTRONICS52119.2021.9422665

[20] M. H. Alwan, M. Singh ja H. F. Mahdi, "Performance comparison of turbo codes with LDPC codes and with BCH codes for forward error correcting codes", teoksessa *2015 IEEE Student Conference on Research and Development (SCOReD)*, 2015, s. 556–560. DOI: 10.1109/SCORED.2015.7449398

[21] G. Prasad, H. A. Latchman, Y. Lee ja W. A. Finamore, "A comparative performance study of LDPC and Turbo codes for realistic PLC channels", teoksessa *18th IEEE International Symposium on Power Line Communications and Its Applications*, 2014, s. 202–207. DOI: 10.1109/ISPLC.2014.6812365

[22] A. V. Naik, H. P, R. K. C ja Y. G. V, "BER Analysis of Channel Coding Technique for 5G", teoksessa *2025 International Conference on Intelligent Computing and Knowledge Extraction (ICICKE)*, 2025, s. 1–6. DOI: 10.1109/ICICKE65317.2025.11136522

[23] B. Tahir, S. Schwarz ja M. Rupp, "BER comparison between Convolutional, Turbo, LDPC, and Polar codes", teoksessa *2017 24th International Conference on Telecommunications (ICT)*, 2017, s. 1–7. DOI: 10.1109/ICT.2017.7998249

[24] "On the Road to 6G: Drivers, Challenges and enabling Technologies", Fraunhofer Institute for Integrated Cirucuits IIS, tekninen raportti, 2021. url: https://www.iis.fraunhofer.de/content/dam/iis/en/doc/ks/bb/6g-sentinel-white-paper.pdf

[25] M. Geiselhart, F. Krieg, J. Clausius, D. Tandler ja S. ten Brink, "6G: A Welcome Chance to Unify Channel Coding?", *IEEE BITS the Information Theory Magazine*, vol. 3, nro 1, s. 67–80, 2023. DOI: 10.1109/MBITS.2023.3322974