



ΠΑΝΕΠΙΣΤΗΜΙΟ
ΘΕΣΣΑΛΙΑΣ

ΤΜΗΜΑ ΨΗΦΙΑΚΩΝ ΣΥΣΤΗΜΑΤΩΝ

ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

**«ΣΥΓΚΡΙΣΗ ΤΕΧΝΙΚΩΝ ΚΩΔΙΚΟΠΟΙΗΣΗΣ ΜΕ ΧΡΗΣΗ
OPEN SOURCE SOFTWARE»**

ΣΚΟΥΜΑ ΑΛΕΞΑΝΔΡΑ

ΕΠΙΒΛΕΠΩΝ ΚΑΘΗΓΗΤΗΣ: ΧΑΪΚΑΛΗΣ ΚΩΝΣΤΑΝΤΙΝΟΣ

ΛΑΡΙΣΑ 2024

THESIS TITLE:

**"COMPARISON OF CODING TECHNIQUES USING
OPEN SOURCE SOFTWARE"**

ΕΥΧΑΡΙΣΤΙΕΣ

Θα ήθελα να ευχαριστήσω την αγαπημένη μου οικογένεια που πάντα με βοηθούσε και με στήριζε καθ' όλη τη διάρκεια των σπουδών μου στο Πανεπιστήμιο Θεσσαλίας.

Στην οικογένειά μου

ΠΕΡΙΛΗΨΗ

Στην παρούσα πτυχιακή εργασία επιτεύχθηκε η μαζικότερη και πιο συσσωρευμένη συγκέντρωση πληροφοριών σχετικά με την Σύγκριση τεχνικών κωδικοποίησης με χρήση open source software.

Σε ένα πρώτο επίπεδο στο Κεφάλαιο 1, έγινε μία ανάλυση της Εισαγωγής προσπαθώντας να επιτευχθεί μία προλογική ανάπτυξη στο τι θα ακολουθήσει.

Στο Κεφάλαιο 2, επιχειρήθηκε η συγκεντρωτική κειμενική ανάπτυξη η οποία σχετίζεται με την εφαρμόσιμη μεθοδολογία έρευνας.

Στο Κεφάλαιο 3, πραγματοποιήθηκε η βιβλιογραφική ανασκόπηση πάνω στο κύριο θέμα της εργασίας.

Εν κατακλείδι, στο Κεφάλαιο 4, παρατέθηκαν συμπεράσματα τα οποία με την σειρά τους έρχονται να εδραιώσουν την τελική και ολοκληρωμένη άποψη και εικόνα του ενδότερου κειμενικού περιεχομένου, στοχεύοντας στην παρουσίαση της προσωπικής, αντικειμενικής και παράλληλα συμπερασματικής άποψης πάνω στο αναπτυσσόμενο θέμα που προσεγγίστηκε στην κειμενική εξέλιξη του παρόντος πονήματος.

Λέξεις – Κλειδιά: Τεχνικές Κωδικοποίησης, Open Source Software

ABSTRACT

In this thesis, the largest and most accumulated collection of information regarding the Comparison of coding techniques using open source software was achieved.

At a first level in Chapter 1, an analysis of the Introduction was done trying to achieve a prologue development to what will follow.

In Chapter 2, the integrated textual development related to the applied research methodology is attempted.

In Chapter 3, the bibliographic review was carried out on the main topic of the paper.

In conclusion, in Chapter 4, conclusions were presented which in turn come to establish the final and comprehensive view and image of the inner textual content, aiming at the presentation of the personal, objective and at the same time conclusive view on the developing topic that was approached in the textual evolution of the present pain.

Keywords: Coding Techniques, Open Source Software

ΠΕΡΙΕΧΟΜΕΝΑ

ΠΕΡΙΛΗΨΗ.....	5
ABSTRACT.....	6
ΠΕΡΙΕΧΟΜΕΝΑ.....	7
Κατάλογος Σχημάτων.....	9
Κατάλογος Πινάκων.....	11
Κεφάλαιο 1: Εισαγωγή.....	12
Κεφάλαιο 2: Μεθοδολογία Έρευνας.....	15
Κεφάλαιο 3: Ανασκόπηση Βιβλιογραφίας.....	16
3.1 Συνελικτικοί Κώδικες.....	16
3.1.1 Παράδειγμα Συνελικτικού Κώδικα.....	16
3.1.2 Τρόποι Αναπαράστασης Συνελικτικού Κώδικα.....	18
3.2 BLOCK Κώδικες.....	20
3.2.1 Παράδειγμα BLOCK κώδικα.....	22
3.3 TURBO Κώδικας.....	25
3.3.1 TURBO Κωδικοποίηση.....	27
3.3.2 TURBO Αποκωδικοποίηση.....	29
3.3.3 INTERLEAVER Σχεδίαση.....	37
3.3.4 TURBO Κώδικες σε Συστήματα 3G.....	39
3.3.5 TURBO Κώδικες σε Συστήματα 4G.....	42
3.3.6 AFF3CT Προσομοίωση.....	43
3.4 POLAR Κώδικας.....	45
3.4.1 Τεχνική Πόλωσης Καναλιού.....	46
3.4.2 Κωδικοποίηση κατά ARIKAN.....	49
3.4.3 Αποκωδικοποίηση κατά ARIKAN.....	53
3.4.4 Πλεονέκτηματα – Μειονεκτήματα.....	53
3.4.5 Ανεπτυγμένη Έκδοση SCD.....	54
3.4.6 Κώδικας PAC.....	54
3.4.7 Μελλοντικές Επεκτάσεις.....	56
3.5 LDPC Κώδικας.....	56
3.5.1 Τρόποι Αναπαράστασης LDPC Κώδικα.....	57
3.5.2 Κωδικοποίηση LDPC.....	59
3.5.3 Αποκωδικοποίηση LDPC.....	62
3.5.4 Εφαρμογές.....	68
3.5.5 Πλεονεκτήματα -Μειονεκτήματα LDPC.....	69

3.6 AFF3CT	70
3.6.1 OUTPUT	70
3.6.2 TURBO Προσομοίωση	72
3.6.3 POLAR Προσομοίωση	74
3.6.4 LDPC Προσομοίωση	76
3.7 Συγκρίσεις Τεχνικών Κωδικοποίησης	77
3.7.1 TURBO vs POLAR vs LDPC	77
Κεφάλαιο 4: Συμπεράσματα	85
Βιβλιογραφία	86

Κατάλογος Σχημάτων

Σχήμα 1: Βασικό σύστημα επικοινωνίας. Πηγή: Bernard, 2005	12
Σχήμα 2: Κατηγορίες FEC. Πηγή: Ιδία Επεξεργασία	14
Σχήμα 3: Κέρδος Κωδικοποίησης. Πηγή: Ιδία Επεξεργασία.....	14
Σχήμα 4: Επίπεδα Μελετητικής Μεθοδολογίας. Πηγή: Ιδία Επεξεργασία.....	15
Σχήμα 5: Συνελκτικός κώδικας. Πηγή: Branka et al., 2010.....	17
Σχήμα 6: Συνελκτικός κώδικας. Πηγή: Branka et al., 2010.....	17
Σχήμα 7: Διάγραμμα Καταστάσεων. Πηγή: Ιδία Επεξεργασία	18
Σχήμα 8: Διάγραμμα Δέντρου. Πηγή: Ιδία Επεξεργασία	19
Σχήμα 9: Διάγραμμα Trellis. Πηγή: Ιδία Επεξεργασία.....	20
Σχήμα 10: γραμμικός κώδικας με κωδικολέξεις. Πηγή: Ιδία Επεξεργασία	22
Σχήμα 11: Πίνακας Γεννήτορας. Πηγή: Ιδία Επεξεργασία.....	23
Σχήμα 12: Πίνακας Γεννήτορας. Πηγή: Ιδία Επεξεργασία.....	24
Σχήμα 13: Block κώδικας. Πηγή: Ιδία Επεξεργασία	24
Σχήμα 14: Γραμμικός Κώδικας. Πηγή: Ιδία Επεξεργασία	25
Σχήμα 15: Παράδειγμα ενός RSC κωδικοποιητή. Πηγή: Ιδία Επεξεργασία	28
Σχήμα 16: Παράδειγμα ενός non-RSC κωδικοποιητή. Πηγή: Ιδία Επεξεργασία	28
Σχήμα 17: Κωδικοποίηση Turbo. Πηγή: Ιδία Επεξεργασία	29
Σχήμα 18: Αλγόριθμος MAP. Πηγή: Ιδία Επεξεργασία	31
Σχήμα 19: Block διάγραμμα, διαδικασία της αποκωδικοποίησης ενός turbo κώδικα. Πηγή: Ιδία Επεξεργασία.....	35
Σχήμα 20: Αναλυτική δομή ενός turbo αποκωδικοποιητή. Πηγή: Ιδία Επεξεργασία	36
Σχήμα 21: Παράδειγμα ενός Turbo κώδικα. Πηγή: Ιδία Επεξεργασία.....	38
Σχήμα 22: Αποτέλεσμα προσομοίωσης (FEC) για το σύστημα WCDMA. Πηγή: https://www.researchgate.net/	41
Σχήμα 23: Αποτέλεσμα προσομοίωσης (FEC) για το σύστημα CDMA-2000. Πηγή: https://www.researchgate.net/	41
Σχήμα 24: Δυναμικός κώδικας Turbo. Πηγή: Ιδία Επεξεργασία.....	42
Σχήμα 25: Δυναμικός κώδικας Turbo. Πηγή: Ιδία Επεξεργασία.....	43
Σχήμα 26: συστηματικός Turbo κώδικας (3084.1024). Πηγή: https://imtech.imt.fr	44
Σχήμα 27: Συνδυασμός καναλιών. Πηγή: Ιδία Επεξεργασία.....	46
Σχήμα 28: Παράδειγμα πόλωσης. Πηγή: https://www.computerworld.com/	48
Σχήμα 29: Κωδικοποίηση Polar για Rank=8. Πηγή: https://www.computerworld.com/	50
Σχήμα 30: Σύγκριση Κέρδους σε systematic, non-systematic. Πηγή: https://www.computerworld.com/	51
Σχήμα 31: Κέρδος FER σε systematic. Πηγή: https://www.computerworld.com/	52
Σχήμα 32: Ροή διαδικασίας και εξαγωγή δεδομένων. Πηγή: Ιδία Επεξεργασία	55
Σχήμα 33: Subcode κόμβοι. Πηγή: https://www.dsprelated.com/	58
Σχήμα 34: τμήμα γραφήματος κώδικα LDPC. Πηγή: Ιδία Επεξεργασία	60
Σχήμα 35: Turbo (18444.1644) και i=11. Πηγή: https://aff3ct.github.io/	72
Σχήμα 37: Σύγκριση επιδόσεων κώδικα Turbo για N=18444 και N=3084. Πηγή: https://aff3ct.github.io/	73
Σχήμα 38: Polar (2048.1723) για i=15. Πηγή: https://aff3ct.github.io/	74
Σχήμα 39: Σύγκριση επιδόσεων κώδικα Polar για N=1024 και N=4096. Πηγή: https://aff3ct.github.io/	75
Σχήμα 40: Απόδοση αλγορίθμων NMS και SPA για LDPC(16200.11880). Πηγή: https://aff3ct.github.io/	76

Σχήμα 41: K 128,576,8000. Πηγή: https://aff3ct.github.io/	77
Σχήμα 42: Απόδοση περιοχής και ενέργειας στους διάφορους αποκωδικοποιητές καναλιών. Πηγή: https://aff3ct.github.io/	79
Σχήμα 43: Τρόπος με τον οποίο επηρεάζεται η απόδοση σε συνάρτηση με τον αριθμό των επαναλήψεων για Turbo και LDPC και για το μέγεθος λίστας του Polar. Πηγή: https://aff3ct.github.io/	80
Σχήμα 44: Απόδοση των σχημάτων κωδικοποίησης. Πηγή: https://aff3ct.github.io/	81
Σχήμα 45: N 4096, 4108, 4224 και R=1/2. Πηγή: https://aff3ct.github.io/	82
Σχήμα 46: Σύγκριση με διαφορετικό K,N,R,i. Πηγή: https://aff3ct.github.io/	83

Κατάλογος Πινάκων

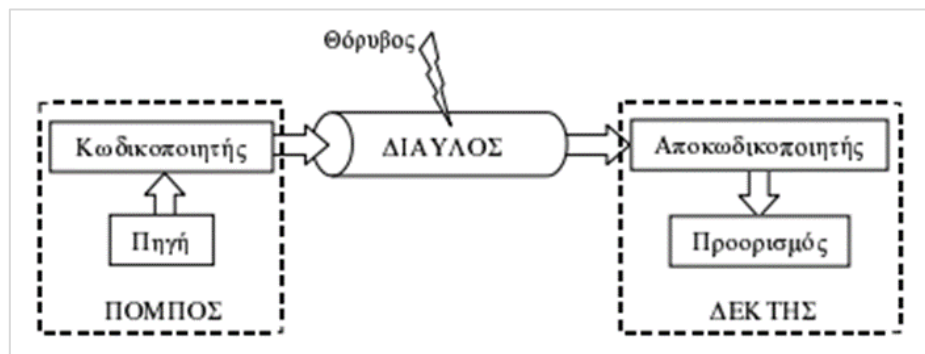
Πίνακας 1: Παράδειγμα ενός interleaver. Πηγή: Ιδία Επεξεργασία	37
--	----

Κεφάλαιο 1: Εισαγωγή

Στη σημερινή εποχή, η ανάπτυξη και η εξέλιξη της τεχνολογίας και της επικοινωνίας έχει προχωρήσει ραγδαία. Αυτό έχει ως αποτέλεσμα να αυξάνονται οι απαιτήσεις της κοινωνίας για καλύτερες και όσο το δυνατόν ισχυρότερες τεχνικές. Έτσι, η επιστήμη έχει κεντρίσει το ενδιαφέρον της προς την βελτίωση των ήδη υπάρχον τεχνικών κωδικοποίησης και αποκωδικοποίησης πάνω στον τομέα της Πληροφορικής (Bernard, 2005).

Έως το 1948, οι τηλεπικοινωνιακοί μηχανικοί υποστήριζαν ότι για να βελτιωθεί η ποιότητα και η ταχύτητα της επικοινωνίας είναι απαραίτητο να μειωθεί ο ρυθμός μετάδοσης της πληροφορίας, συνεπώς, να περιοριστεί ο αριθμός των σφαλμάτων του μηνύματος. Το έναυσμα λοιπόν, για την δημιουργία της θεωρία της πληροφορίας και της κωδικοποίησης αποτέλεσε το άρθρο ορόσημο του Αμερικανού Claude Elwood Shannon. Ο ίδιος διατύπωσε το θεώρημα Noisy Channel Coding (θεώρημα θορυβώδους καναλιού) όπου όρισε τη χωρητικότητα του καναλιού ως άνω φράγμα του ρυθμού μετάδοσης. Επίσης, όρισε την εντροπία υποστηρίζοντας ότι εκφράζει ένα μαθηματικό όριο που σχετίζεται με την ακεραιότητα των δεδομένων της πηγής σε ένα κανάλι με σχεδόν μηδενικό θόρυβο (Branka et al., 2010).

Πάνω από τον μέγιστο ρυθμό μετάδοσης ενός επικοινωνιακού καναλιού είναι αδύνατον να υπάρχει επικοινωνία χωρίς απώλεια δεδομένων. Όμως, μπορεί να υπάρξει κάτω από το όριο με πιθανότητα σφάλματος όσο μικρή (ακόμη και μηδενική) επιθυμεί ο σχεδιαστής του συστήματος. Απέδειξε ότι είναι εφικτή η βελτιστοποίηση τόσο της αξιοπιστίας όσο και του ρυθμού μετάδοσης ταυτόχρονα και η πιθανότητα σφαλμάτων να τείνει προς το μηδέν. Όλα αυτά, μπορούν να συμβούν εφόσον τα δεδομένα έχουν υποστεί την απαραίτητη κωδικοποίηση. Αυτό επιτυγχάνεται προσθέτοντας στα αρχικά δεδομένα πλεονάζουσα πληροφορία προκειμένου να καταπολεμηθεί ο θόρυβος (Bernard, 2005).



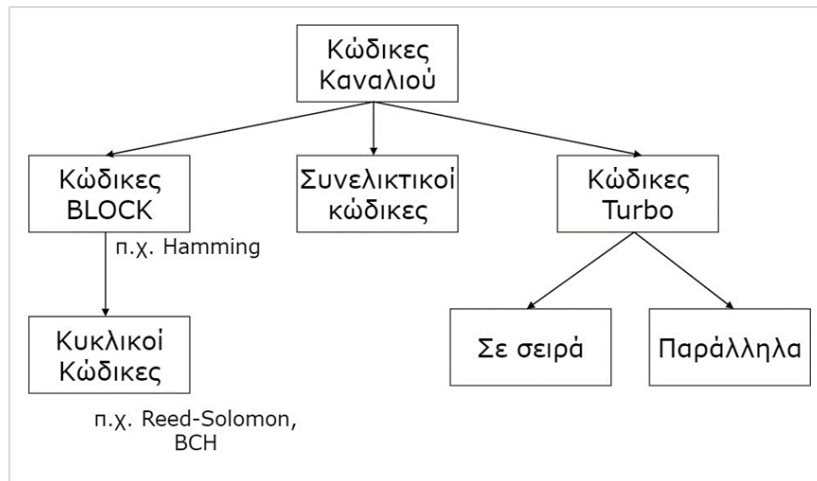
Σχήμα 1: Βασικό σύστημα επικοινωνίας. Πηγή: Bernard, 2005

Με τον όρο κώδικα, γίνεται αναφορά σε ένα σύνολο από κανόνες προκειμένου να μετατραπεί ένα κομμάτι πληροφορίας σε διαφορετική μορφή. Η θεωρία, λοιπόν, των κωδικών αποτελείται από όλους αυτούς τους κανόνες που ορίζουν με ποιον τρόπο θα εισαχθεί πλεονάζουσα πληροφορία, δηλαδή bit, έτσι ώστε να επιτευχθεί η μετάδοση της πληροφορίας και η ανίχνευση σφαλμάτων. Έτσι, οι μέθοδοι κωδικοποίησης καναλιών εμπίπτουν γενικά σε δύο κατηγορίες: κωδικούς μπλοκ ή ισομήκεις (block codes) και συνελκτικούς κώδικες (convolutional codes) (Branka et al., 2010).

Οι κωδικοί μπλοκ λειτουργούν σε ένα μπλοκ δεδομένων/bit με σταθερό μέγεθος και εφαρμόζουν τη χειραγώγηση σε αυτό το μπλοκ στον πομπό και τον δέκτη. Οι κώδικες Reed-Solomon, που χρησιμοποιούνται συνήθως στους σκληρούς δίσκους των υπολογιστών, είναι ένα παράδειγμα αυτού του τύπου κώδικα. Οι συνελκτικοί κώδικες, από την άλλη πλευρά, λειτουργούν σε ροές δεδομένων με πιο αυθαίρετους αριθμούς δεδομένων/bit. Για αυτό, σημαντική παραγωγική έρευνα έχει γίνει όλα αυτά τα χρόνια για τη συνένωση των μπλοκ και των συνελκτικών κωδικών για να συνδυαστούν τα οφέλη και των δύο.

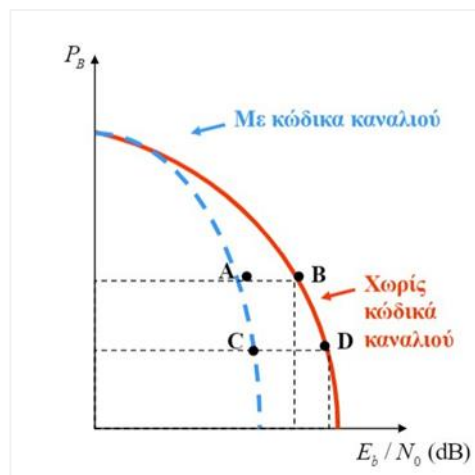
Η κωδικοποίηση καναλιού (channel coding) αναφέρετε στη διαδικασία εισαγωγής, πριν την μετάδοση, πλεονάζουσας πληροφορίας προκειμένου να εντοπιστούν και να διορθωθούν σφάλματα που δημιουργήθηκαν κατά την μετάδοση της πληροφορίας. Τα ενδεχόμενα λάθη προκύπτουν από την προσθήκη θορύβου (μεταβολές ή αλλοιώσεις) που υφίσταται το σήμα από εξωγενείς παράγοντες. Η διόρθωση λοιπόν, των λαθών γίνεται με τους εξής τρόπους (MacKay et al., 1996):

- Την μέθοδο ARQ (automatic repeat request ή αίτημα αυτόματης μετάδοσης). Περιλαμβάνει αμφίδρομη επικοινωνία, δηλαδή όταν εντοπιστεί σφάλμα στον δέκτη ζητείται εκ νέου η μετάδοση, των δεδομένων που χάθηκαν, από τον πομπό.
- Την μέθοδο FEC (forward error correction ή διόρθωση σφαλμάτων προώθησης). Είναι ουσιαστικά η μετάδοση περιττής πληροφορίας μαζί με την βασική ροή πληροφορίας προκειμένου ο δέκτης να έχει την δυνατότητα να εντοπίσει και να διορθώσει λάθη μετάδοσης χωρίς να επαναληφθεί η διαδικασία και να αποκατασταθούν τα πακέτα που χάθηκαν. Οι κυριότερες κατηγορίες FEC κωδικών εμφανίζονται στην παρακάτω εικόνα:



Σχήμα 2: Κατηγορίες FEC. Πηγή: Ιδία Επεξεργασία

Αποτέλεσμα της κωδικοποίησης καναλιού είναι η ελάχιστη απαίτηση ισχύς εκπομπής ώστε να πετύχει την ίδια πιθανότητα σφάλματος. Η μείωση ισχύος ονομάζεται Κέρδος Κωδικοποίησης και μετριέται σε dB. Η μετάδοση της έξτρα πληροφορίας μειώνει το ρυθμό μετάδοσης της χρήσιμης πληροφορίας (MacKay et al., 1996).



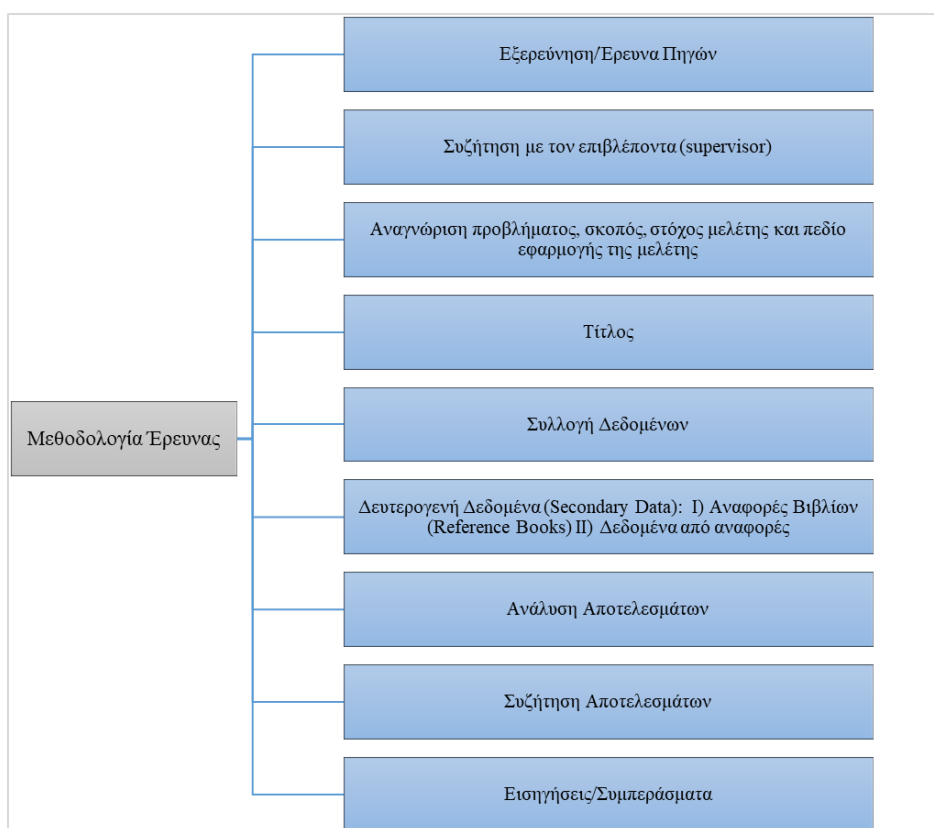
Σχήμα 3: Κέρδος Κωδικοποίησης. Πηγή: Ιδία Επεξεργασία

Κεφάλαιο 2: Μεθοδολογία Έρευνας

Η μελετητική μεθοδολογία βασίζεται στην αναλυτική στοχοθέτηση της μελέτης. Πρόκειται για μια εκτενή περιγραφή και έκθεση του κύριου θέματος " Σύγκριση τεχνικών κωδικοποίησης με χρήση open source software" ως γενικό τίτλο. Η μελέτη σχετίζεται με ανασκόπηση βιβλιογραφίας.

Επιπλέον, παρέχεται μια πιο εξειδικευμένη ανάλυση των δεδομένων από τις μελέτες που αναπτύχθηκαν και ακολουθεί ένα τμήμα συμπερασμάτων, συζήτησης και τρόπων βελτίωσης.

Η αναζήτηση διενεργήθηκε μέσω της χρήσης του διαδικτύου, συγγραμμάτων που σχετίζονται με το ληφθέν θέμα επεξεργασία.



Σχήμα 4: Επίπεδα Μελετητικής Μεθοδολογίας. Πηγή: Ιδία Επεξεργασία

Κεφάλαιο 3: Ανασκόπηση Βιβλιογραφίας

3.1 Συνελικτικοί Κώδικες

Μία σημαντική κατηγορία κωδικών στις Τηλεπικοινωνίες αποτελούν οι συνελικτικοί κώδικες. Ανακαλύφθηκαν το 1954 από τον Peter Elias ενώ στη θεωρία τους εμβάθυνε ο Forney. Το όνομα τους οφείλεται στον τρόπο που γίνεται η κωδικοποίηση της πληροφορίας, καθώς τα ψηφία τους εφαρμόζονται σε δύο συναρτήσεις, δηλαδή με συνέλιξη και με τη βοήθεια απόκρουσης ενός καταχωρητή ολίσθησης. Οι συγκεκριμένοι κώδικες διαφέρουν σε σχέση με τους block κώδικες, κυρίως, στα στοιχεία μνήμης που διαθέτουν. Οι συνελικτικοί σε κάθε χρονική στιγμή, η έξοδος εξαρτάται και από τις προηγούμενες καταστάσεις του και όχι μόνο από την τρέχουσα είσοδο (Bernard, 2005).

Χαρακτηρίζονται από τρεις παραμέτρους (n, k, m), όπου το n είναι ο αριθμός των bits εξόδου, το k είναι ο αριθμός των bits εισόδου και τέλος το m όπου είναι ο αριθμός των στοιχείων μνήμης. Επιπρόσθετα, αποτελούνται και από τον ρυθμό R , όπου ορίζεται από τον αριθμό των bits εισόδου προς τον αριθμό των bits εξόδου, δηλαδή, $R = k/n$ και από το μήκος περιορισμού L όπου είναι ο αριθμός των bits της μνήμης που επηρεάζουν την παραγωγή των n bits εξόδου και υπολογίζεται $L = k(m-1)$ (Bernard, 2005).

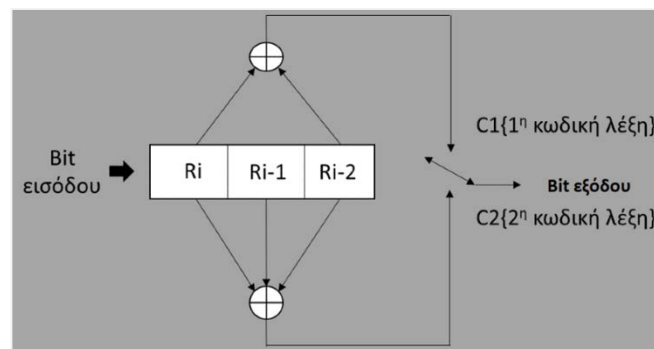
Υλοποιείται με ένα καταχωρητή ολίσθησης και από το modulo-2 αθροιστές και διαθέτει μία διάταξη με μνήμη διότι πρέπει να αναγνωρίζει και τις προηγούμενες καταστάσεις που έχουν περάσει για να καταλήξει στο επιθυμητό αποτέλεσμα. Τέλος, οι συνελικτικοί κώδικες μπορούν να αναπαρασταθούν με τρεις γραφικούς τρόπους, με το διάγραμμα κατάστασεων, με δένδρο και με το διάγραμμα Trellis και μπορούν να αποκωδικοποιηθούν με τον αλγόριθμο Fano και τον αλγόριθμο Viterbi (Branka et al., 2010).

3.1.1 Παράδειγμα Συνελικτικού Κώδικα

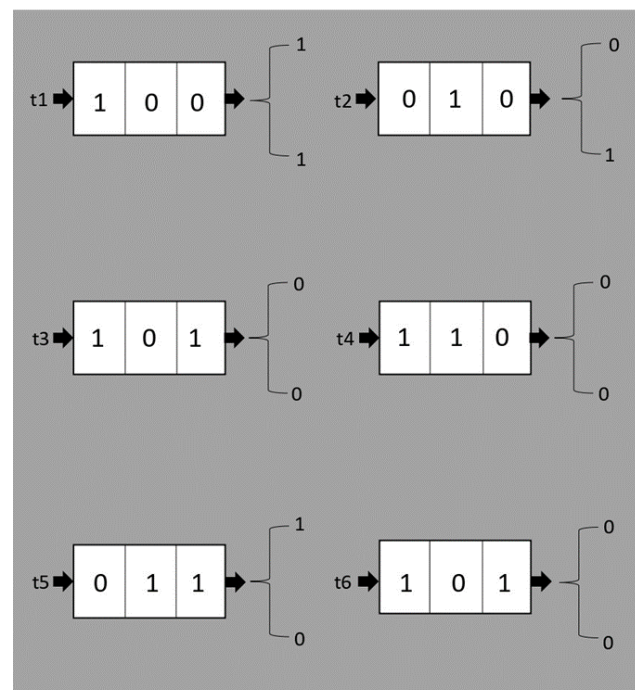
Οι συνελικτικοί κώδικες κωδικοποιούν τα δεδομένα τους συνεχώς για αυτό και ανήκουν στην κατηγορία των κωδικών Trellis. Η κωδικοποίηση τους πραγματοποιείται με μία απλή διαδικασία. Τα στοιχεία που χρησιμοποιούνται στον κώδικα είναι ένας καταχωρητής ολίσθησης (Shift-Register), όπου στην πρώτη θέση υπάρχει το bit εισόδου και στις υπόλοιπες θέσεις τα δεδομένα bits από τη μνήμη του συνελικτικού κωδικοποιητή m καθώς και οι αθροιστές modulo-2 που πραγματοποιούνται μόνο με την πύλη Boolean XOR. Ο συνελικτικός κώδικας παριστάνεται από μία ακολουθία γεννητριών g ενώ ισοδύναμα αναπαρίστανται ως G . Πριν ξεκινήσει το πρώτο bit, ο κωδικοποιητής περιέχει μόνο μηδενικά στοιχεία εκτός και αν προοριστεί διαφορετικά. Αφού ξεκινήσει η διαδικασία, τα bits πληροφορίας εισέρχονται στους

καταχωρητές μνήμης, ένα k bit εισόδου κάθε φορά και n bits στην έξοδο του, τα οποία στέλνονται για μετάδοση. Αφού τελειώσει η πρώτη διαδικασία, το επόμενο bit που θα εισέλθει στον κωδικοποιητή θα μετακινήσει όλες τις τιμές του καταχωρητή δεξιά και θα περιμένει το επόμενο bit εισόδου. Η διαδικασία θα συνεχιστεί μέχρι να καταχωρηθεί και το τελευταίο k bit στον κωδικοποιητή και αφού έχει γίνει η αποστολή των τελευταίων bits εξόδου (Bernard, 2005).

Στο παρακάτω σχήμα απεικονίζεται ένας συνελκτικός κώδικας ο οποίος έχει ρυθμό $R = \frac{1}{2}$ και $L = 3$ ($m=2$). Τα πολυώνυμα γεννητόρας είναι $g_1 = [101]$ και $g_2 = [111]$. Το μήνυμα εισόδου θα είναι: $[101101]$. Συμπερασματικά λοιπόν, η έξοδος του συνελκτικού κώδικα είναι: $[11\ 01\ 00\ 00\ 10\ 00]$ (Branka et al., 2010).



Σχήμα 5: Συνελκτικός κώδικας. Πηγή: Branka et al., 2010



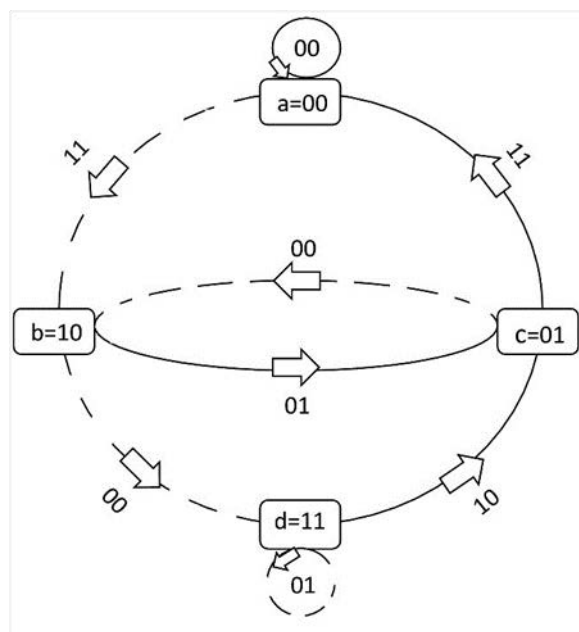
Σχήμα 6: Συνελκτικός κώδικας. Πηγή: Branka et al., 2010

3.1.2 Τρόποι Αναπαράστασης Συνελικτικού Κώδικα

3.1.2.1 Διάγραμμα Καταστάσεων

Στο παρακάτω σχήμα απεικονίζεται ο συνελκτικός κώδικας $n,1,k$ ο οποίος μπορεί να περιγραφεί από ένα διάγραμμα καταστάσεων εφόσον είναι ένα κύκλωμα με πεπερασμένες καταστάσεις. Εξαρτάται από το περιεχόμενο που έχει η μνήμη του m στη συγκεκριμένη περίπτωση 2 και τον συνολικό αριθμό καταστάσεων όπου είναι 2^m . Αποτελείται από 3 καταχωρητές ολίσθησης αλλά το διάγραμμα καταστάσεων αξιοποιεί μόνο τα δύο τελευταία στάδια ($k2,k3$). Από τη στιγμή που αναγνωρίζει την τρέχουσα κατάσταση του κωδικοποιητή, το ψηφίο που εισέλθετε αυτόματα αναγνωρίζει και την έξοδο αλλά και την επόμενη κατάσταση (Bernard, 2005).

Η κατάσταση του αλλάζει μόνο όταν γίνει ολίσθηση μηνύματος του κωδικοποιητή. Αποτελείται από κόμβους και βέλη τα οποία δείχνουν την τωρινή και την επόμενη κατάσταση. Το 1 παριστάνεται με διακεκομμένη γραμμή και το 0 με συνεχή γραμμή και χρησιμοποιούνται όταν γίνει η μετάβαση από τη μία κατάσταση στην άλλη. Πάνω στις γραμμές τοποθετούνται οι έξοδοι των ψηφίων (n) και αποτελείται από τέσσερις καταστάσεις τις 00,01,10,11. Βασικό μειονέκτημα του διαγράμματος κατάστασης είναι ότι δεν απεικονίζει τον χρόνο οπότε είναι άγνωστο σε ποια χρονική στιγμή βρίσκονταν ο κώδικας (Branka et al., 2010).



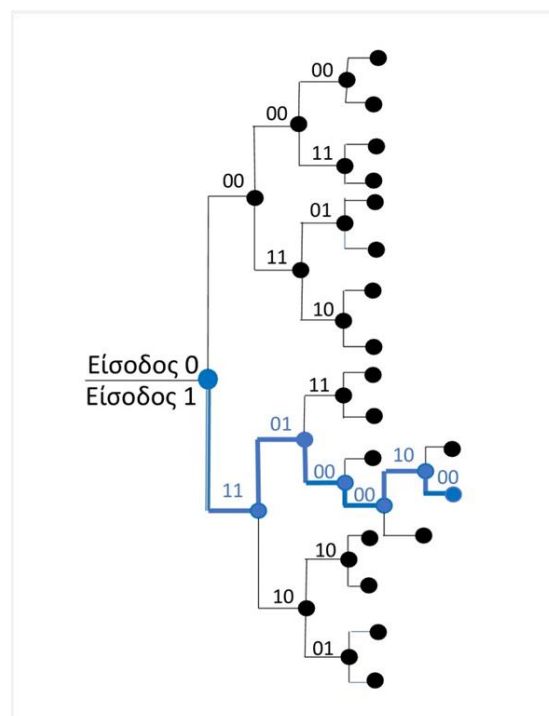
Σχήμα 7: Διάγραμμα Καταστάσεων. Πηγή: Ιδία Επεξεργασία

3.1.2.2 Διάγραμμα Δέντρου

Το διάγραμμα δέντρου είναι ο δεύτερος τρόπος αναπαράστασης ενός συνελκτικού κώδικα και είναι παρόμοιος με το διάγραμμα καταστάσεων, με την κύρια διαφορά ότι το συγκεκριμένο

διάγραμμα δείχνει και την εξέλιξη του κωδικοποιητή στο χρόνο. Καθορίζεται από την είσοδο ενός δυαδικού ψηφίου κάθε φορά που εισέρχεται στον καταχωρητή, με αποτέλεσμα να δημιουργείται το διάγραμμα με ανάλογο τρόπο ξεκινώντας από τα αριστερά προς τα δεξιά. Η έξοδος του κωδικοποιητή φαίνεται από το κάθε κλαδί του δέντρου ενώ κάθε κόμβος δείχνει την κατάσταση που είναι το σύστημα, δηλαδή αν το δυαδικό ψηφίο εισόδου είναι 0, τότε το κλαδί μετακινείται δεξιά και προς τα πάνω, ενώ αν το δυαδικό ψηφίο είναι 1 τότε το κλαδί μετακινείται δεξιά και προς τα κάτω (Bernard, 2005).

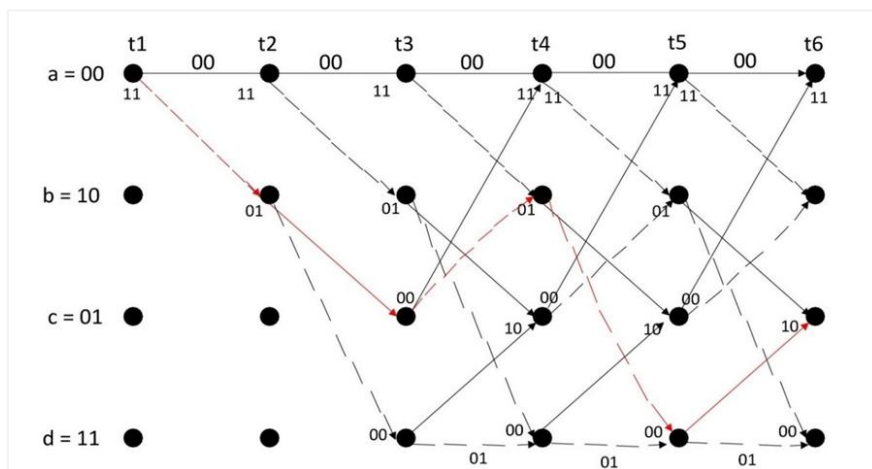
Η είσοδος του συγκεκριμένου συνελκτικού κωδικοποιητή αποτελείται από τα εξής δυαδικά ψηφία: 101101. Η πορεία που θα ακολουθήσει το πρώτο κλαδί του δέντρου θα είναι δεξιά και προς τα κάτω με έξοδο 11 που αναγράφεται πάνω στο ίδιο κλαδί. Με τον ίδιο ακριβώς τρόπο συνεχίζεται και η πορεία των επόμενων δυαδικών ψηφίων εισόδου. Ένα μειονέκτημα που έχει το διάγραμμα δέντρου είναι ότι μεγαλώνει πολύ τον αριθμό των κλαδιών του με συνάρτηση 2^L όπου το L είναι όλοι οι πιθανοί έξοδοι που θα μπορούσε να έχει ο κωδικοποιητής, με αποτέλεσμα να υπάρχει μεγάλος όγκος κλαδιών και περισσότερες προϋποθέσεις ώστε να ολοκληρωθεί το διάγραμμα, εφόσον είναι απαραίτητο να αναγράφονται όλες οι περιπτώσεις του (Bernard, 2005).



Σχήμα 8: Διάγραμμα Δέντρου. Πηγή: Ιδία Επεξεργασία

3.1.2.3 Διάγραμμα TRELLIS

Ο τρίτος και τελευταίος τρόπος αναπαράστασης ενός συνελκτικού κώδικα είναι το διάγραμμα Trellis. Το διάγραμμα Trellis είναι συνδυασμός από το διάγραμμα καταστάσεων και δέντρου και αποτελεί τον καλύτερο και τον πιο αποτελεσματικό τρόπο σε σύγκριση με τους δύο προηγούμενους. Το σχεδιάγραμμα του καθορίζεται από το διάγραμμα καταστάσεων καθώς μπορεί να κατευθύνει όλες τις πιθανές διαδρομές εισόδου και εξόδου. Τα βέλη κατευθύνονται ανάλογα με το χρόνο και έχουν την αρμοδιότητα να μεταβιβάζουν μία κατάσταση σε μια άλλη. Κάτι ακόμη που καθορίζεται από το χρόνο, είναι το διάγραμμα το οποίο μετακινείται προς τη δεξιά κατεύθυνση και αυτόματα γίνεται και η μεταβίβαση ενός καινούργιου δυαδικού ψηφίου από τον πίνακα. Όταν γίνει είσοδος με δυαδικό ψηφίο 1, εμφανίζεται διακεκομμένη γραμμή, ενώ όταν γίνει με 0, εμφανίζεται με ευθεία γραμμή. Ξεκινάει από την κατάσταση $a = 00$ και όταν εισέλθει το πρώτο bit από την είσοδο, στη συγκεκριμένη περίπτωση το 1, τότε θα γίνει μεταβίβαση στη κατάσταση $b = 10$ ενώ αν ήταν το 0 τότε θα γινόταν μεταβίβαση στην κατάσταση 00. Ο κώδικας θα φτάσει στον μεγαλύτερο αριθμό καταστάσεων δηλαδή 2^m μετά από K χρονικές περιόδους και μετά θα τερματιστεί η διαδικασία (Branka et al., 2010).



Σχήμα 9: Διάγραμμα Trellis. Πηγή: Ιδία Επεξεργασία

3.2 BLOCK Κώδικες

Σύμφωνα με τις ιστοσελίδες <https://www.icpsr.umich.edu/>, <https://eclass.uoa.gr/>, <https://www.telecomreview.com/> και <https://www.researchgate.net/>, δύνανται να ειπωθούν οι παρακάτω αναφορές σχετικά με τους κώδικες Block.

Οι block κώδικες ανακαλύφθηκαν το 1950 από τον Αμερικανό μαθηματικό Richard Hamming. Στην αρχή, ήταν να πάρουν την ονομασία "Hamming" από το επίθετο του εφευρέτη τους αλλά τελικά ονομάστηκαν block κώδικες. Το βασικό και κύριο χαρακτηριστικό

που έχουν οι block κώδικες είναι ότι δίνουν τη δυνατότητα στο χρήστη να εντοπίσει και να ανιχνεύσει πιθανά σφάλματα του κώδικα χωρίς να επικοινωνήσει με την πηγή του κώδικα. Ουσιαστικά, οι block κώδικες είναι σε θέση να βοηθήσουν τον παραλήπτη από την αρχή της κωδικοποίησης και να διορθώσουν έναν περιορισμένο αριθμό σφαλμάτων, με αποτέλεσμα η συγκεκριμένη ενέργεια να αποφύγει την αναμετάδοση του μηνύματος η οποία μπορεί να διαρκέσει χρόνο. Υπάρχουν πολλοί τύποι από block κώδικες οι οποίοι είναι οι: κώδικες Reed-Solomon, κώδικες Hamming, κωδικοί επέκτασης, κώδικες Hadamard, κωδικοί Golay, κώδικες Reed-Muller. Οι συγκεκριμένοι κώδικες είναι γνωστοί ως κυκλικοί ή αλγεβρικοί κώδικες block επειδή μπορούν να δημιουργηθούν χρησιμοποιώντας πολυώνυμα boole. Ωστόσο, η αποκωδικοποίηση τους είναι πολύ δύσκολη διότι χρησιμοποιούνται αλγεβρικοί αποκωδικοποιητές. Οι block κώδικες είναι κωδικοποιητές χωρίς μνήμη για αυτό και εξαρτώνται μόνο από τις τιμές του τρέχοντος μηνύματος, σε σχέση με τους συνελκτικούς κώδικες που έχουν μνήμη. Μπορούν να αναφερθούν σε οποιονδήποτε κωδικό διόρθωσης σφαλμάτων που ενεργεί σε έναν block κώδικα των k bits δεδομένων εισόδου για παραγωγή n bits δεδομένων εξόδου (n,k) . Στην είσοδο του κωδικοποιητή καναλιού τύπου block έχουμε μία λέξη δεδομένων των k bit και προσθέτοντας ο κωδικοποιητής καναλιού $n-k$ bit ελέγχου ισοτιμίας, εξάγεται μια κωδικολέξη των n bit. Στους block κώδικες η ακολουθία των ψηφίων, δηλαδή το μήνυμα, αποτελεί την κωδική λέξη όπου τα k bit της λέξης δεδομένων βρίσκονται είτε στην αρχή της κωδικολέξης είτε στο τέλος ($n-k$ αριστερή, k bit δεξιά) και ονομάζονται συστηματικοί. Τέλος, οι block κώδικες ονομάζονται και γραμμικοί διότι το αποτέλεσμα της πρόσθεσης με modulo-2 αριθμητική, με οποιεσδήποτε δύο κωδικές λέξεις είναι μια άλλη κωδικολέξη του κώδικα, όπως επίσης και η κωδική λέξη περιέχει μόνο μηδενικά ψηφία. Η πρώτη εφαρμογή ενός γραμμικού block κώδικα έγινε το 1979-1981 με ένα διαστημόπλοιο που έστειλε έγχρωμες φωτογραφίες από τους πλανήτες Δία και Άρη χρησιμοποιώντας κώδικα. Σε κάθε 4096 κωδικές λέξεις υπήρχε και μία διαφορετική απόχρωση ηλιακού συστήματος. Ο κώδικας αυτός είναι γνωστός ως Golay Γ24.

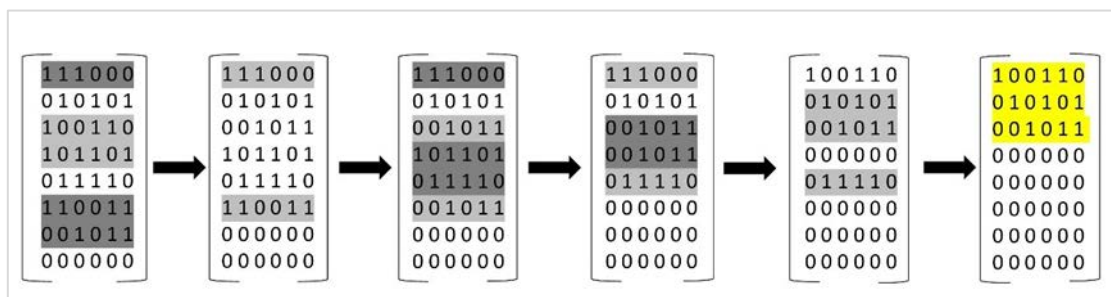


Εικόνα 7. Κωδικολέξη. Πηγή: <https://www.telecomreview.com/>

3.2.1 Παράδειγμα BLOCK κώδικα

Όπως αναφέρθηκε και παραπάνω, οι block κώδικες ονομάζονται γραμμικοί όταν το αποτέλεσμα οποιονδήποτε δύο κωδικολέξεων είναι μία άλλη κωδικολέξη του κώδικα και με αυτόν τον τρόπο μπορεί να βρεθεί μία βάση βρίσκοντας το ελάχιστο πλήθος γραμμικώς ανεξάρτητων κωδικολέξεων. Η ελάχιστη απόσταση Hamming ενός γραμμικού block κώδικα προκύπτει από την κωδικολέξη με το ελάχιστο βάρος Hamming χωρίς την κωδικολέξη του μηδενικού διανύσματος. Μία βάση ενός γραμμικού block κώδικα μπορεί να βρεθεί τοποθετώντας όλες τις κωδικολέξεις στις γραμμές ενός πίνακα, μετατρέποντας τον πίνακα σε περιορισμένης κλιμακωτής διάταξης γραμμών. Ο πίνακας μετατρέπεται σε κλιμακωτής διάταξης γραμμών με ανταλλαγή μεταξύ γραμμών και με αντικατάσταση μιας γραμμής από το αποτέλεσμα της άθροισης της ίδιας με κάποια/ες άλλη/ες.

Έστω γραμμικός κώδικας με κωδικολέξεις: (111000), (010101), (100110), (101101), (011110), (110011), (001011), (000000).



Σχήμα 10: γραμμικός κώδικας με κωδικολέξεις. Πηγή: Ιδία Επεξεργασία

- Στον πρώτο πίνακα έγινε δυαδική πρόσθεση του C1: 111000 + C6: 110011 + C7: 001011 = 000000 που πήρε θέση στον δεύτερο πίνακα με C7 και C3: 100110 + C4: 101101 = 001011 στο δεύτερο πίνακα με C3.
- Στο δεύτερο πίνακα έγινε δυαδική πρόσθεση του C1: 111000 + C6: 110011 = 001011 που πήρε θέση στον τρίτο πίνακα με C6.
- Στον τρίτο πίνακα έγινε δυαδική πρόσθεση του C1: 111000 + C4: 101101 + C5: 011110 = 001011 που πήρε θέση στον τέταρτο πίνακα με C4 και C3: 001011 + C6: 001011 = 000000 στον τέταρτο πίνακα με C6.
- Στον τέταρτο πίνακα έγινε δυαδική πρόσθεση του C1: 111000 + C5: 011110 = 100110 που πήρε θέση στον πέμπτο πίνακα με C1 και C3: 001011 + C4: 001011 = 000000 στον πέμπτο πίνακα με C4.
- Στον πέμπτο πίνακα έγινε δυαδική πρόσθεση του C2: 010101 + C3: 001011 + C5: 011110 = 000000 που πήρε θέση στον έκτο πίνακα με C5.

Οι κωδικολέξεις (100110), (010101), (001011) αποτελούν μία βάση του υπόχωρου με διάσταση 3, επειδή αποτελείται από 3 γραμμές. Κάθε πίνακας που έχει ως γραμμές μία βάση ενός κώδικα ονομάζεται πίνακας γεννήτορας (γεννήτρια) του κώδικα και συμβολίζεται με G .

$$\begin{bmatrix} 1 & 0 & 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \rightarrow G = \begin{bmatrix} 1 & 0 & 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 & 1 & 1 \end{bmatrix}$$

Σχήμα 11: Πίνακας Γεννήτορας. Πηγή: Ιδία Επεξεργασία

Ο πίνακας $G = [I_3|P]$, όπου G ($k \times n$), I ($k \times k$) και P ($k \times m$) ο οποίος είναι και ο πίνακας συντελεστών με τιμές $\{0,1\}$. Η διάσταση του πίνακα G είναι $k \times n = 3 \times 6$, έτσι και ο κώδικας block είναι ο $(n, k) = (6, 3)$. Ο κώδικας ικανοποιεί και το όριο Hamming εφόσον $n \leq 2^m - 1 = 6 \leq 2^3 - 1 = 7$, όπου $m = n - k$. Το n είναι το μήκος, το k η διάσταση και το $R = k/n$ είναι ο ρυθμός του κώδικα οπότε $R = 1/2$.

Το διάνυσμα μιας λέξης δεδομένων αναπαρίστανται με $d = \{d_1, d_2, \dots, d_k\}$ και το διάνυσμα της αντίστοιχης κωδικολέξης με $c = \{c_1, c_2, \dots, c_n\}$ για αυτό και η κάθε κωδικολέξη ενός γραμμικού block κώδικα σχηματίζεται από τον πίνακα γεννήτορα G ως: $c = d * G$. Με πολλαπλασιασμό της κάθε κωδικής λέξης δεδομένων $d=3$ με τον G προκύπτουν οι αντίστοιχες κωδικολέξεις c . Ο πίνακας γεννήτορας δίνει απευθείας τις κωδικές λέξεις $[x_1 \ x_2 \ x_3] \ G = [x_1 \ x_2 \ x_3 \ x_4 \ x_5 \ x_6]$.

Προκύπτει δηλαδή:

$$\begin{array}{ll}
1. \begin{pmatrix} 000 \end{pmatrix} G = \begin{pmatrix} 000 \end{pmatrix} \begin{pmatrix} 100 & 110 \\ 010 & 101 \\ 001 & 011 \end{pmatrix} = \begin{pmatrix} 000000 \end{pmatrix} & 2. \begin{pmatrix} 001 \end{pmatrix} G = \begin{pmatrix} 001 \end{pmatrix} \begin{pmatrix} 100 & 110 \\ 010 & 101 \\ 001 & 011 \end{pmatrix} = \begin{pmatrix} 001011 \end{pmatrix} \\
3. \begin{pmatrix} 010 \end{pmatrix} G = \begin{pmatrix} 010 \end{pmatrix} \begin{pmatrix} 100 & 110 \\ 010 & 101 \\ 001 & 011 \end{pmatrix} = \begin{pmatrix} 010101 \end{pmatrix} & 4. \begin{pmatrix} 011 \end{pmatrix} G = \begin{pmatrix} 011 \end{pmatrix} \begin{pmatrix} 100 & 110 \\ 010 & 101 \\ 001 & 011 \end{pmatrix} = \begin{pmatrix} 011110 \end{pmatrix} \\
5. \begin{pmatrix} 100 \end{pmatrix} G = \begin{pmatrix} 100 \end{pmatrix} \begin{pmatrix} 100 & 110 \\ 010 & 101 \\ 001 & 011 \end{pmatrix} = \begin{pmatrix} 100110 \end{pmatrix} & 6. \begin{pmatrix} 101 \end{pmatrix} G = \begin{pmatrix} 101 \end{pmatrix} \begin{pmatrix} 100 & 110 \\ 010 & 101 \\ 001 & 011 \end{pmatrix} = \begin{pmatrix} 101101 \end{pmatrix} \\
7. \begin{pmatrix} 110 \end{pmatrix} G = \begin{pmatrix} 110 \end{pmatrix} \begin{pmatrix} 100 & 110 \\ 010 & 101 \\ 001 & 011 \end{pmatrix} = \begin{pmatrix} 110011 \end{pmatrix} & 8. \begin{pmatrix} 111 \end{pmatrix} G = \begin{pmatrix} 111 \end{pmatrix} \begin{pmatrix} 100 & 110 \\ 010 & 101 \\ 001 & 011 \end{pmatrix} = \begin{pmatrix} 111000 \end{pmatrix}
\end{array}$$

Σχήμα 12: Πίνακας Γεννήτορας. Πηγή: Ιδία Επεξεργασία

Άρα, θα δημιουργηθεί ο πίνακας:

Λέξη Δεδομένων	Κωδικολέξη
000	000000
001	001011
010	010101
011	011110
100	100110
101	101101
110	110011
111	111000

Σχήμα 13: Block κώδικας. Πηγή: Ιδία Επεξεργασία

Ο block κώδικας είναι συστηματικός, διότι τα 3 πρώτα bit της κάθε κωδικολέξης είναι η λέξη δεδομένων.

Επιπλέον, το πλήθος μη μηδενικών ψηφίων bits μίας κωδικής λέξης ενός κώδικα ονομάζεται βάρος c . Σημαντικό ρόλο παίζει και η ελάχιστη απόσταση μεταξύ των κωδικών λέξεων d του κώδικα όπου ισχύει το εξής, όσο μεγαλύτερο είναι το d τόσα πιο πολλά σφάλματα μπορεί να αναγνωρίσει. Η ελάχιστη απόσταση d είναι ίση με το ελάχιστο μη μηδενικό βάρος των κωδικών λέξεων του κωδικά. Οπότε, έχει τη δυνατότητα να ανιχνεύσει μέχρι $d-1$ κωδικά σφάλματα και αν d είναι περιττός αριθμός τότε μπορεί $(d-1)/2$ σφάλματα αλλά αν d άρτιος αριθμός τότε διορθώνει $d/2-1$ σφάλματα. Στο συγκεκριμένο παράδειγμα, η ελάχιστη απόσταση του κώδικα είναι $d_{\min}=3$ και μπορεί να διορθώσει έως ένα λάθος ανά κωδικολέξη.

2^{ος} τρόπος:

Πίνακας ελέγχου ισοτιμίας $H = [P|I_3]$ είναι ανάστροφος πίνακας G . Ο πίνακας P έχει διαστάσεις 3×3 , ενώ πίνακας H έχει 3×6 , το $k=3$ και $n=6$. Το αρχικό μήνυμα έχει 3 ψηφία και θα προστεθούν άλλα 3 πλεονάζοντα ψηφία για να γίνει η κωδική λέξη.

Στον πίνακα ελέγχου ισοτιμίας πρέπει να ισχύει το εξής:

$$H^* \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ \vdots \\ x_k \end{bmatrix} = 0$$

Οπότε:

$$H \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \end{bmatrix} = 0 \iff \left(\begin{array}{ccc|ccc} 1 & 1 & 0 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 \\ 0 & 1 & 1 & 0 & 0 & 1 \end{array} \right) \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \end{bmatrix} = 0 \iff \begin{bmatrix} x_1 + x_2 + x_4 \\ x_1 + x_3 + x_5 \\ x_2 + x_3 + x_6 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}$$

Τα x_1, x_2, x_3 είναι γνωστά και θα γίνει ο υπολογισμός των x_4, x_5, x_6 για να βρεθεί ο παρακάτω γραμμικός κώδικας.

x_1	x_2	x_3	x_4	x_5	x_6
0	0	0	0	0	0
0	0	1	0	1	1
0	1	0	1	0	1
0	1	1	1	1	0
1	0	0	1	1	0
1	0	1	1	0	1
1	1	0	0	1	1
1	1	1	0	0	0

Σχήμα 14: Γραμμικός Κώδικας. Πηγή: Ιδία Επεξεργασία

3.3 TURBO Κώδικας

Σύμφωνα με τις ιστοσελίδες <https://www.researchgate.net/>, <https://imtech.imt.fr/>, <http://www.scholarpedia.org/>, <https://slideplayer.com/> και <https://www.researchgate.net/>, δύνανται να ειπωθούν οι παρακάτω αναφορές σχετικά με τον κώδικα Turbo.

Για πολλά χρόνια επιστήμονες και ερευνητές προσπαθούσαν να ανακαλύψουν μία τεχνική κωδικοποίησης όπου θα πλησίαζε τη βέλτιστη απόδοση και θα προσέγγιζε στενά το όριο Shannon ή αλλιώς μέγιστο κανάλι χωρητικότητας. Στην αρχή, έγιναν πολλά πειράματα σε πολλούς κώδικες όπως block, συνελκτικούς, BCH αλλά κανένας θεωρητικά δεν ήταν κοντά στα αποτελέσματα τους. Το 1991, λοιπόν, αποδείχθηκε ότι μόνο μία κατηγορία τεχνικών κωδικοποίησης μπορεί να πετύχει τη βέλτιστη απόδοση. Ο μοναδικός εφευρέτης που κατάφερε να την ανακαλύψει ήταν ο Claude Berrou. Η πρώτη δημοσίευση του κώδικα έγινε το 1993 με όνομα: “Κωδικοποίηση και αποκωδικοποίηση διόρθωσης σφαλμάτων κοντά στο όριο Shannon: Κωδικοί στροβίλου” από τρεις ερευνητές τον Claude Berrou, Alain Glavieux, Punya Thitimajshima, στη Γαλλία. Ονομάστηκαν κώδικες Turbo και ήταν τόσο ισχυροί που έκαναν πολλούς ερευνητές και επιστήμονες να μην πιστεύουν στα αναφερόμενα αποτελέσματά του.

Οι ερευνητές του, όμως, υποστήριζαν ότι ο συγκεκριμένος κώδικας θα μπορούσε να διπλασιάζει την απόδοση των δεδομένων για μία συγκεκριμένη ισχύ μετάδοσης ή και να πετύχει έναν συγκεκριμένο ρυθμό μετάδοσης δεδομένων επικοινωνίας με το μισό της ενέργειας που μεταδίδεται. Οι κώδικες Turbo ανήκουν στην κατηγορία των κωδικών διόρθωσης σφαλμάτων (ECC) μαζί με μία τεχνική επαναληπτικής αποκωδικοποίησης και αποδεικνύονται οι περισσότεροι αποδοτικοί κώδικες FEC. Τα σχήματα στους κώδικες Turbo μπορούν να συνδυάσουν και να χρησιμοποιήσουν γνωστούς κώδικες όπως block και συνελκτικούς.

Χρησιμοποιούνται σε εφαρμογές χαμηλής ισχύος όπως δορυφορικές συνδέσεις και επικοινωνίες. Πολλοί ερευνητές έχουν επικεντρωθεί στην συγκεκριμένη τεχνική και έχουν βελτιώσει τη χρήση του για ψηφιακό ήχο και μετάδοση δεδομένων σε δίκτυα Wi-fi. Εκατομμύρια άνθρωποι χρησιμοποιούν πολλές και διάφορες μορφές Turbo εφόσον έχουν ενσωματωθεί και στο σύστημα κινητής τηλεφωνίας.

Το συγκεκριμένο σύστημα κωδικοποίησης έχει τη δυνατότητα να διαχειρίζεται τα δεδομένα πολυμέσων όπως τις εικόνες και τα βίντεο μέσα από φορητές συσκευές και κινητά τηλέφωνα με αποτέλεσμα να ελέγχει κανάλια με θόρυβο. Με τόσο εντυπωσιακή απόδοση που δείχνουν να έχουν οι κώδικες Turbo δε θα μπορούσαν να αφήσουν απαρατήρητη την κοινότητα των Τηλεπικοινωνιών.

Στη σημερινή εποχή, βρίσκονται στην πρώτη γραμμή της έρευνας με τις δύο μεγαλύτερες εταιρείες Τηλεπικοινωνιών στο χώρο, την France Telecom και την NTT DoCoMo σε συνεργασία με άλλα πανεπιστήμια και ομάδες εργασιών. Άλλες εταιρείες επίσης είναι η Sony,

η Samsung, η Motorola, η NEC που αποτελούν οι πιο σημαντικές στην Τηλεπικοινωνία αλλά και νέες εξιδεικευμένες εταιρείες όπως η Turbocash, Turboconcept, iCoding και τέλος οι Broadcom, Conexant, Altera, Xilinx οι οποίες είναι κατασκευαστικές εταιρείες hardware και τσιπ.

3.3.1 TURBO Κωδικοποίηση

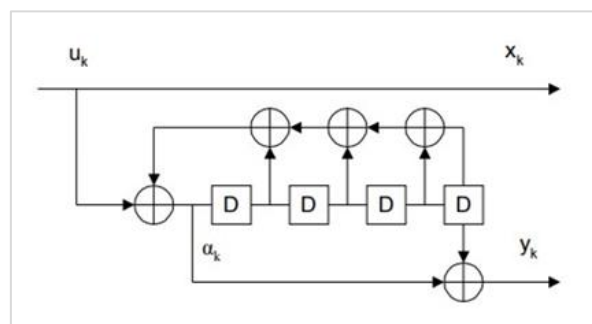
Ένας κωδικοποιητής turbo αποτελείται από δύο συνελκτικούς κώδικες συνδεδεμένους παράλληλα. Ο τρόπος αυτός ονομάζεται παράλληλη αλυσιδωτή κωδικοποίηση (parallel concatenation). Υπάρχει όμως και ένας άλλος τρόπος αλυσιδωτής κωδικοποίησης ο οποίος συνδέει τους συνελκτικούς κώδικες σε σειρά (serial concatenation). Ο αριθμός και το είδος των επιμέρους κωδικών μπορεί να διαμορφωθεί με διάφορους τρόπους, όπως να χρησιμοποιήσει τρεις ή περισσότερους κώδικες αναλόγως την περίπτωση που αναλογεί. Αυτό που αποδεικνύεται όμως είναι ότι αν γίνει χρήση περισσότερων από δύο επιμέρους κωδικοποιητές θα δημιουργηθούν ισχυρότεροι και καλύτεροι turbo κώδικες (multiple turbo codes). Στους turbo κώδικες έχουν επικρατήσει ως επιμέρους στοιχεία του, οι αναδρομικοί συστηματικοί συνελκτικοί κώδικες (RSC, Recursive Systematic Convolution) αν και μπορούν να χρησιμοποιηθούν οποιασδήποτε μορφής κωδικοποιητή. Συστηματικός εννοείτε όταν μία από τις εξόδους του κωδικοποιητή αποτελεί την είσοδο του και σε κάθε κωδική λέξη που περιέχονται k bits πληροφορίας τα $n-k$ bit αποτελούν τα parity bits (bits ισοτιμίας). Ουσιαστικά, τα k bits όταν στέλνονται άμεσα θα είναι υπό επεξεργασία. Συνήθως, χρησιμοποιούνται συστηματικοί κώδικες διότι έχουν την ικανότητα να απλοποιούν την αποκωδικοποίηση χωρίς να επηρεάσουν την απόδοση που θα πετύχει ο turbo κώδικας ενώ ένας μη συστηματικός κώδικας θα μείωνε το ρυθμό μετάδοσης. Επίσης, είναι αναδρομικός διότι μπορεί να χρησιμοποιεί στην είσοδο του κώδικα, μία ή περισσότερες από τις εξόδους του, με τη βοήθεια ενός βρόχου ανάδρασης. Η είσοδος του καταχωρητή ολίσθησης του κώδικα δεν θα είναι ίδια με τα bits πληροφορίας που αναφέρθηκε παραπάνω αλλά θα δοθεί μία άλλη μεταβλητή που θα αντιστοιχεί στη σχέση:

$$a_k \sum_{i=1}^{k-1} g_i = a_{k-1} ,$$

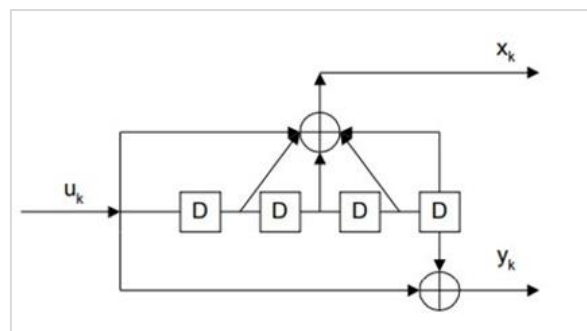
όπου το g θα παίρνει είτε τιμή 0 είτε 1 ανάλογα με το αν θα υπάρχει σύνδεση με την θέση μνήμης στον καταχωρητή ολίσθησης ή αν δεν υπάρχει καθόλου.

Έρευνες μέχρι στιγμής, έχουν δείξει ότι οι RSC κώδικες έχουν παραμείνει ως επιμέρους στοιχεία στον turbo κωδικοποιητή. Οι κώδικες turbo έχουν την δυνατότητα να παράγουν χαμηλά ποσοστά σφάλματος σε χαμηλά SNR (Signal Noise Ratio). Το SNR εκφράζει το λόγο

της ισχύς του σήματος προς την ισχύ θορύβου. Εφόσον συμβαίνει αυτό, επικρατούν κατά αυτόν τον τρόπο οι RSC κώδικες διότι έχουν καλύτερη συμπεριφορά στις χαμηλές SNR σχέσεις, σε αντίθεση με τους non-RSC. Ένας ακόμη λόγος που χρησιμοποιούνται RSC κώδικες είναι για την πληροφορία με μικρό βάρος Hamming, οι οποίοι μπορούν να δώσουν στην έξοδο τους, κωδικές λέξεις πολύ μεγάλου βάρους. Αυτό οφείλεται στο βρόχο ανάδρασης που έχουν για βοηθό, με αποτέλεσμα όταν βρεθεί μία μονάδα στην ακολουθία εισόδου να ανακυκλώνεται συνεχώς, προκαλώντας μία κωδική λέξη με πολύ μεγάλο βάρος, σε αντίθεση με τους non-RSC κώδικες που η επίδραση μιας μονάδας υφίσταται για τόσες χρονικές στιγμές όσο και η μνήμη του κώδικα.



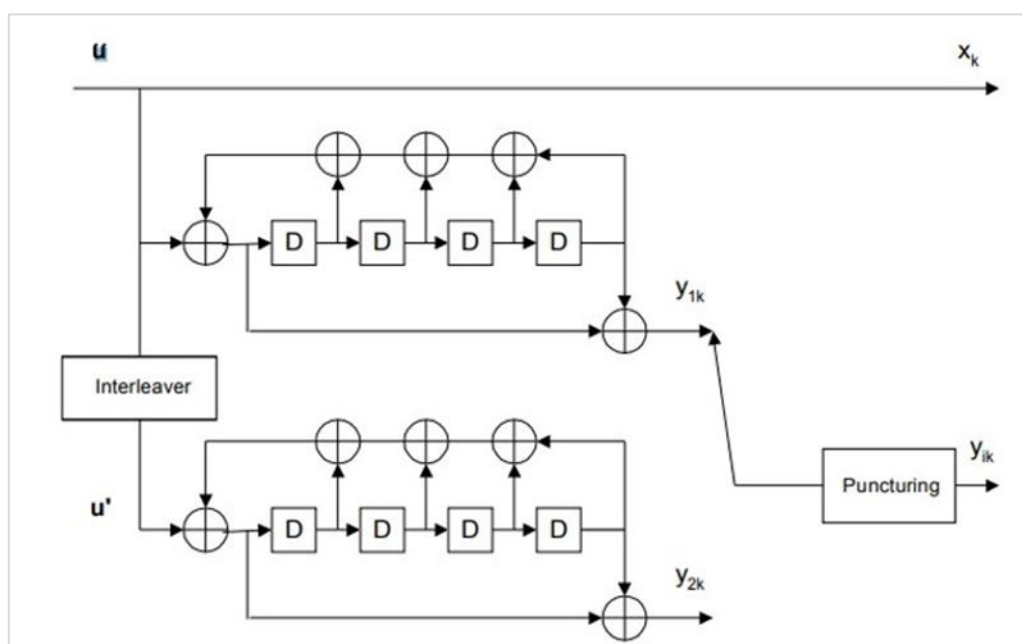
Σχήμα 15: Παράδειγμα ενός RSC κωδικοποιητή. Πηγή: Ιδία Επεξεργασία



Σχήμα 16: Παράδειγμα ενός non-RSC κωδικοποιητή. Πηγή: Ιδία Επεξεργασία

Στο παρακάτω σχήμα απεικονίζεται ένας turbo κωδικοποιητής που αποτελείται από δύο όμοιους RSC κώδικες. Στον πρώτο κωδικοποιητή, υπάρχει μία λειτουργία απευθείας από την ακολουθία των bits πληροφορίας $u_k = (u_1, u_2, \dots, u_n)$ και έχει δύο ακολουθίες εξόδων x_k, y_{1k} . Στον δεύτερο κωδικοποιητή, η λειτουργία του είναι ίδια με την ακολουθία εισόδου αφού πρώτα θα περάσει από την συσκευή αναδιάταξης interleaver. Η δουλειά ενός interleaver είναι να λαμβάνει την ακολουθία των bits πληροφορίας αλλά να την εξάγει με διαφορετική σειρά, σύμφωνα με κάποιον προκαθορισμένο κανόνα. Ουσιαστικά, δέχεται τα ίδια bits αλλά με διαφορετική σειρά και παράγει μία άλλη ακολουθία εξόδου την y_{2k} . Επίσης, υπάρχει και ένας άλλος «κομμένος» κώδικας puncturing ο οποίος χρησιμοποιείται για να αυξήσει το ρυθμό του

κώδικα όπου κάποια bit εξόδου δε στέλνονται με κάποιο κανόνα. Δεν είναι απαραίτητο όμως να τοποθετείται σε όλους του turbo κώδικες. Στην ουσία ένας puncturing ελέγχει τον ρυθμό μετάδοσης του κώδικα, δηλαδή τη μία χρονική στιγμή επιτρέπει την εκπομπή του bit ισοτιμίας του πρώτου κωδικοποιητή με έξοδο y_{1k} και την επόμενη στιγμή τα bit ισοτιμίας του δεύτερου κωδικοποιητή με έξοδο y_{2k} . Άρα, υπάρχει ένας κώδικας με ρυθμό $\frac{1}{2}$ όπου οι επιμέρους κώδικες έχουν μνήμη $M=4$ και οι έξοδοι του δίνονται από τις σχέσεις $x_k=u$, $y_{1k}=u \cdot g_a/g_b$, $y_{2k}=u' \cdot g_a/g_b$, όπου τα πολυώνυμα θα είναι $g_a=(10001)$ και $g_b=(11111)$. Τέλος, η σχεδίαση των επιμέρους κωδικοποιητών αποτελεί σημαντικό παράγοντα για την συμπεριφορά των turbo κωδικών.



Σχήμα 17: Κωδικοποίηση Turbo. Πηγή: Ιδία Επεξεργασία

3.3.2 TURBO Αποκωδικοποίηση

3.3.2.1 Αλγόριθμοι SOFT INPUT ΚΑΙ SOFT OUTPUT

Όπως αναφέρθηκε και σε προηγούμενο κεφάλαιο, η κωδικοποίηση καναλιού διατρέχει συγκεκριμένες ενέργειες ώστε να γίνει σωστά και αποδοτικά η μετάδοση της πληροφορίας. Συγκεκριμένα, η ενέργεια που εκτελείται από τη λήψη και φτάνει στην αποκωδικοποίηση καναλιού, ισοδυναμεί με την απόφαση που πρέπει να γίνει μεταξύ των δύο τιμών 0 ή 1, εφόσον εκτελείται μια hard απόφαση στον αποδιαμορφωτή που βρίσκεται ενδιάμεσα της ενέργειας. Ονομάζεται έτσι, διότι η πληροφορία που θα φτάσει στον αποκωδικοποιητή δεν είναι καθόλου αξιόπιστη για τα αποτελέσματά της. Για αυτό, έχουν αναπτυχθεί αλγόριθμοι αποκωδικοποίησης που μπορούν να δέχονται soft τιμές, στην είσοδο τους αλλά και στην έξοδο τους, ώστε να τις στείλουν στην είσοδο ενός άλλου αποκωδικοποιητή. Ο όρος hard και soft

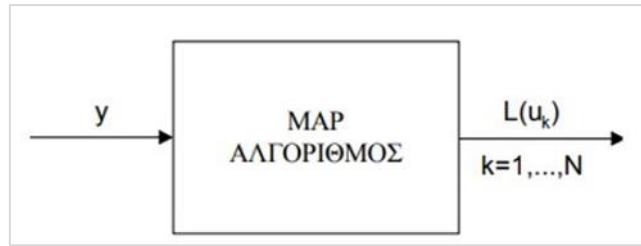
μπορεί να ερμηνευτεί ως εξής: Όταν χρησιμοποιούνται soft τιμές ο αποκωδικοποιητής περιέχει περισσότερη πληροφορία συγκριτικά με τις hard τιμές για αυτό και έχουν καλύτερη απόδοση, όμως η υλοποίηση του αλγορίθμου θα είναι περίπλοκη. Ένα παράδειγμα που θα βοηθήσει στην κατανόηση του αλγορίθμου είναι να υπάρχουν δύο πιθανότητες $\pi(1)=0,7$ και $\pi(0)=0,9$ αντίστοιχα. Οι τιμές αυτές ονομάζονται soft τιμές και η τιμή που μπορεί να πάρει το ένα bit είναι 0 ή 1. Αν τελικά το bit θα πάρει την τιμή 0 τότε $\pi(0)>\pi(1)$, η απόφαση που έχει πάρει είναι hard γιατί βρίσκεται πιο μακριά από την τιμή του. Αν το bit έχει την τιμή 0,699 θα πάρει soft απόφαση γιατί θα είναι πιο κοντά στην τιμή 1 και όχι στην τιμή 0.

Όταν ένας αποκωδικοποιητής μπορεί να λαμβάνει soft τιμές στην είσοδο του, δηλαδή, soft input αλλά και στην έξοδο του, δηλαδή, soft output τότε υπάρχει SISO αποκωδικοποιητής. Σε αυτή την περίπτωση, η έξοδος θα έχει πάρει την τιμή που της αναλογεί. Οι κώδικες turbo έχουν ένα τέτοιο είδος τεχνικής αποκωδικοποίησης οδηγώντας σε χαμηλή πιθανότητα σφάλματος. Ένα παράδειγμα SISO αλγορίθμου είναι ο MAP αλγόριθμος.

3.3.2.2 MAP Αλγόριθμος

Ο αλγόριθμος MAP ονομάζεται έτσι από τα αρχικά των λέξεων του Maximum A posteriori Probability όπου στα ελληνικά σημαίνει "Μέγιστη εκ των υστέρων πιθανότητα". Παρουσιάστηκε για πρώτη φορά από τους Bahl, Cocke, Jelinek και Raviv το 1974 και για αυτό η πρώτη ονομασία που είχε ο αλγόριθμος ήταν BCJR από τα αρχικά των ονομάτων τους. Ο αλγόριθμος MAP έχει τη δυνατότητα να βρίσκει το πιθανό bit πληροφορίας που έχει μεταδοθεί εφόσον αναγνωρίζει και την ακολουθία που έχει ληφθεί. Πιο συγκεκριμένα, είναι ένας αλγόριθμος εκτίμησης τυχαίων παραμέτρων με τη βοήθεια προηγούμενων κατανομών. Συνήθως πολλοί συγκρίνουν τον αλγόριθμο MAP με τον αλγόριθμο Viterbi, ο οποίος βρίσκει την πιο πιθανή ακολουθία πληροφορίας που έχει μεταδοθεί για αυτό και οι επιδόσεις των δύο αλγορίθμων δεν είναι πολύ διαφορετικές σε υψηλές θορυβικές σχέσεις. Όμως, σε χαμηλές σχέσεις ο αλγόριθμος MAP είναι πολύ καλύτερος στα αποτελέσματά του αλλά χειρότερος στην πολυπλοκότητα, αφού είναι πολύ πιο σύνθετος. Με τους κώδικες Turbo ο αλγόριθμος MAP ξαναήρθε στο προσκήνιο μαζί με τη βοήθεια της soft πληροφορίας που μπορεί να χρησιμοποιηθεί σε μία τεχνική επαναληπτικής αποκωδικοποίησης. Πριν τους κώδικες Turbo ο αλγόριθμος Viterbi είχε χρησιμοποιηθεί πολύ περισσότερο από τον MAP αλγόριθμο διότι ήταν πιο απλός και έβγαζε σχεδόν τα ίδια αποτελέσματα.

Παρακάτω απεικονίζεται το σχήμα του αλγορίθμου MAP:



Σχήμα 18: Αλγόριθμος MAP. Πηγή: Ιδία Επεξεργασία

Ο αλγόριθμος MAP έχει σκοπό να δώσει το λόγο LLR(log-likelihood ratio) δηλαδή την πιθανότητα $u_k = +1$ προς την πιθανότητα που είναι $u_k = -1$, με την προϋπόθεση να έχει ληφθεί η ακολουθία y . Αυτό που προκύπτει είναι η πράξη:

$$\ln \frac{\sum_{\substack{s', s \\ u_k = +1}} p(s', s, y_1^N) / p(y_1^N)}{\sum_{\substack{s', s \\ u_k = -1}} p(s', s, y_1^N) / p(y_1^N)} =$$

Το u_k είναι το σύμβολο που στάλθηκε τη χρονική στιγμή k και το y_1^N είναι η ακολουθία όλων των N συμβόλων που έχουν ληφθεί από το 1ο έως το N -οστό.

Η εξίσωση αυτή από το θεώρημα του Baye's μπορεί να γραφεί ως εξής:

$$\begin{aligned} L(\hat{u}_k) &= \ln \frac{\sum_{\substack{s', s \\ u_k = +1}} p(s', s / y_1^N)}{\sum_{\substack{s', s \\ u_k = -1}} p(s', s / y_1^N)} = \\ &= \ln \frac{\sum_{\substack{s', s \\ u_k = +1}} p(s', s, y_1^N)}{\sum_{\substack{s', s \\ u_k = -1}} p(s', s, y_1^N)} \end{aligned}$$

Το s' είναι η κατάσταση του κωδικοποιητή πριν την εκπομπή του bit u_k και το s είναι μετά την εκπομπή του bit u_k αντίστοιχα.

Η πιθανότητα $p(s', s, y_1^N)$ προκύπτει από τη σχέση που δίνεται:

$$\begin{aligned}
P(s', s, y_1^N) &= P(s', s, y_1^{k-1}, y_k, y_{k+1}^N) = \\
&= P(y_{k+1}^N / s', s, y_1^{k-1}, y_k) P(s', s, y_1^{k-1}, y_k) = \\
&= P(y_{k+1}^N / s) P(y_k, s / s', y_1^{k-1}) P(s', y_1^{k-1}) = \\
&= P(y_{k+1}^N / s) P(y_k, s / s') P(s', y_1^{k-1}) = \\
&= \beta_k(s) \cdot \gamma_k(s', s) \cdot a_{k-1}(s')
\end{aligned}$$

Άρα, η εξίσωση μπορεί να γραφεί και ως εξής:

$$L(\hat{u}_k) = \ln \frac{\sum_{\substack{s', s \\ u_k = +1}} a_{k-1}(s') \gamma_k(s', s) \beta_k(s)}{\sum_{\substack{s', s \\ u_k = -1}} a_{k-1}(s') \gamma_k(s', s) \beta_k(s)}$$

Το $a_k(s)$ δίνεται από:

$$\begin{aligned}
a_k(s) &= P(s, y_1^k) = \sum_{s'} P(s', s, y_1^k) = \\
&= \sum_{s'} P(s', s, y_1^{k-1}, y_k) = \sum_{s'} P(s, y_k / s', y_1^{k-1}) P(s', y_1^{k-1}) = \\
&= \sum_{s'} P(s, y_k / s') P(s', y_1^{k-1}) = \sum_{s'} \gamma_k(s', s) a_{k-1}(s')
\end{aligned}$$

όπου η πιθανότητα $P(s, y_k / s', y_1^{k-1}) = P(s, y_k / s')$. Σύμφωνα με την ιδιότητα του συστήματος Markov αν είναι γνωστή η κατάσταση $s_k = s'$ τότε μετά από την χρονική στιγμή k , τα γεγονότα δεν θα εξαρτώνται από την ακολουθία y_1^{k-1} .

Το $\beta_k(s)$ δίνεται από:

$$\begin{aligned}
\beta_k(s) &= P(y_{k+1}^N / s) = \sum_{s''} P(s'', y_{k+1}^N / s) = \\
&= \sum_{s''} P(s'', y_{k+1}, y_{k+2}^N / s) = \sum_{s''} \frac{P(s'', s, y_{k+1}, y_{k+2}^N)}{P(s)} \\
&= \sum_{s''} \frac{P(y_{k+2}^N / s'', s, y_{k+1}) P(s'', s, y_{k+1})}{P(s)} = \sum_{s''} P(y_{k+2}^N / s'') P(s'', y_{k+1} / s) \\
&= \sum_{s''} \beta_{k+1}(s'') \gamma_{k+1}(s, s'')
\end{aligned}$$

Για τον υπολογισμό του $\beta_k(s)$ έχει θεωρηθεί ότι το s'' είναι η κατάσταση του κωδικοποιητή για τη χρονική στιγμή $k+1$, αμέσως μετά από την s κατάσταση.

Και τέλος το $\gamma_k(s)$ δίνεται από:

$$\begin{aligned}\gamma_k(s', s, y_k) &= P(y_k, s/s') = \frac{P(y_k, s', s)}{P(s')} = \\ &= \frac{P(y_k/s', s)P(s', s)}{P(s')} = \\ &= P(y_k/s', s)P(s/s')\end{aligned}$$

Συμπερασματικά λοιπόν, από τη σχέση 'α' αυτό που παρατηρείται είναι ότι ο υπολογισμός γίνεται με μία μέθοδο ανατροφοδότησης προς τα εμπρός ενώ από τη σχέση 'β' η μέθοδος ανατροφοδότησης γίνεται προς τα πίσω. Για να λειτουργήσει σωστά όμως ένας αλγόριθμος MAP θα πρέπει όλες οι περιπτώσεις να καταλήγουν στη μηδενική κατάσταση. Για τα $\alpha_k(s)$, $\beta_k(s)$ ισχύουν:

- $\alpha_0(0)=1$ και $\alpha_0(j)=0$ για κάθε $j \neq 0$
- $\beta_N(0)=1$ και $\beta_N(j)=0$ για κάθε $j \neq 0$

Σύμφωνα με τις παραπάνω συνθήκες ο κωδικοποιητής μπορεί να αρχίζει και να τελειώνει με μηδενική κατάσταση.

3.3.3.3 Διαδικασία Αποκωδικοποίησης

Όταν ανακαλύφθηκαν οι κώδικες Turbo οι ερευνητές πρότειναν μαζί και μια τεχνική επαναληπτικής αποκωδικοποίησης βασισμένη στον αλγόριθμο MAP, αφού αποδείχτηκε ότι τα αποτελέσματα της μπορούν να συγκλίνουν με εκείνα της βέλτιστης αποκωδικοποίησης.

Ένας αλγόριθμος MAP δίνει τον λόγο LLR και προκύπτει η εξίσωση:

$$L(\hat{u}_k) = \ln \frac{\sum_{\substack{s', s \\ u_k = +1}} a_{k-1}(s') \gamma_k(s', s) \beta_k(s)}{\sum_{\substack{s', s \\ u_k = -1}} a_{k-1}(s') \gamma_k(s', s) \beta_k(s)}$$

Αν η μεταβίβαση γίνει από την κατάσταση s' στην s τότε η σχέση που δίνεται για τη $\gamma_k(s', s)$ είναι: $\gamma_k(s', s) = P(u_k)P(y_k/u_k)$ όπου είναι δυνατή η εκπομπή του ψηφίου u_k , ενώ, υπό άλλες συνθήκες θα δίνονταν η σχέση όπως αναφέρθηκε στο προηγούμενο κεφάλαιο. Το u_k είναι το σύμβολο που στέλνεται ενώ το y_k είναι η παραλαβή που γίνεται στο δέκτη σε χρονική στιγμή k .

Με τη σχέση $L(u_k) = \ln \frac{P(u_k = +1)}{P(u_k = -1)}$ η πιθανότητα $P(u_k)$ βρίσκεται μέσα από:

$$P(u_k = \pm 1) = \frac{e^{\pm L(u_k)}}{1 + e^{\pm L(u_k)}} = \left(\frac{e^{-L(u_k)}}{1 + e^{-L(u_k)}} \right) \cdot e^{u_k L(u_k)/2} = C_k \cdot e^{u_k L(u_k)/2}$$

Αν γίνει αποστολή του συμβόλου u_k μετά την κωδικοποίηση θα γίνει αντιστοίχιση στο $x_k=(x_{k,s},x_{k,p})$. Οι δείκτες s και p δείχνουν το συστηματικό bit και το bit ισοτιμίας. Εξαιτίας του θορύβου που θα γίνει κατά την λήψη στο κανάλι εμφανίζεται $y_k=(y_{k,s},y_{k,p})$. Για αυτό το λόγο η πιθανότητα $P(y_k,u_k)$ γράφεται ως εξής:

$$\begin{aligned} P(y_k/u_k) &= P(y_{k,s}/u_k)P(y_{k,p}/x_{k,p}) = \\ &= k_s e^{\frac{-(y_{k,s}-u_k)^2}{2\sigma^2}} \cdot k_p e^{\frac{-(y_{k,p}-x_{k,p})^2}{2\sigma^2}} = \\ &= k_s k_p e^{\frac{-(y_{k,s}^2 + u_k^2 + y_{k,p}^2 + x_{k,p}^2)}{2\sigma^2}} \cdot e^{\frac{y_{k,s}u_k + y_{k,p}x_{k,p}}{\sigma^2}} \\ &= D_k e^{\frac{y_{k,s}u_k + y_{k,p}x_{k,p}}{\sigma^2}} \end{aligned}$$

Αν γίνει αντικατάσταση των δύο τελευταίων εξισώσεων που δίνει το $\gamma_k(s',s)$ τότε αυτό που προκύπτει είναι μία άλλη εξίσωση η:

$$\gamma_k(s',s) = C_k D_k e^{\frac{u_k L(u_k)}{2}} e^{\frac{y_{k,s}u_k + y_{k,p}x_{k,p}}{\sigma^2}}$$

Όταν θα γίνει αντικατάσταση του $\gamma_k(s',s)$ στην αρχική εξίσωση δηλαδή με τον αλγόριθμο MAP που δίνει τον λόγο LLR, τότε τα C_k, D_k θα μπορούν να εμφανιστούν και στον αριθμητή και στον παρονομαστή για αυτό μπορούν να παραλειφθούν με αποτέλεσμα να δημιουργείται μία καινούργια εξίσωση:

$$\begin{aligned} \gamma_k(s',s) &\propto \exp \left[\frac{1}{2} u_k \left(L(u_k) + \frac{2}{\sigma^2} y_{k,s} \right) + \frac{1}{2} \frac{2}{\sigma^2} y_{k,p} x_{k,p} \right] = \\ &= \exp \left[\frac{1}{2} u_k (L(u_k) + L_c y_{k,s}) \right] \cdot \gamma_k^c(s',s) \end{aligned}$$

Το $L_c=2/\sigma^2$ και το $\gamma_k^c(s',s) = L_c y_{k,p} x_{k,p}/2$.

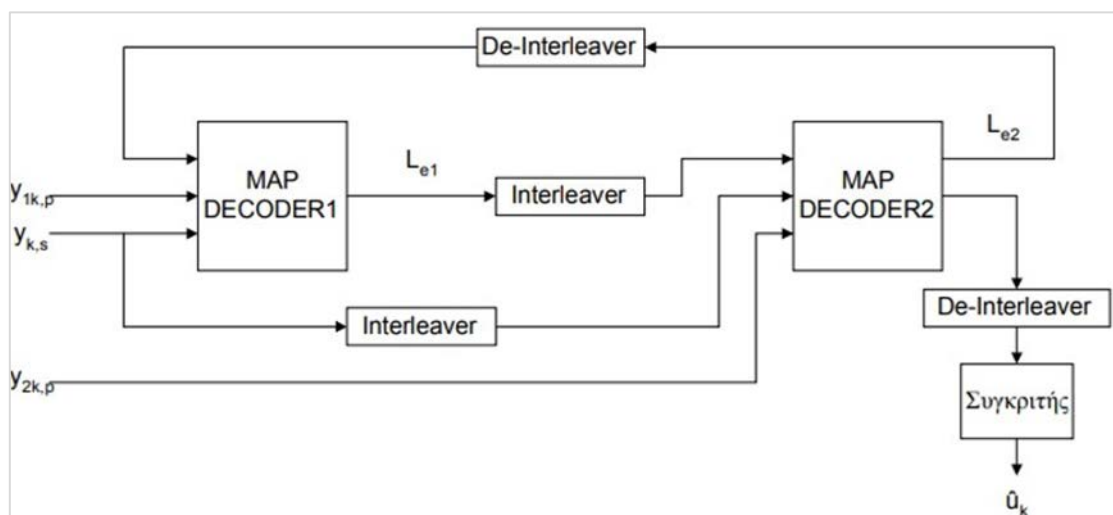
Αφού γίνει αντικατάσταση του $\gamma_k(s',s)$ της αρχικής εξίσωσης, επειδή το πρώτο εκθετικό με την τελευταία εξίσωση θα είναι κοινό για όλους τους όρους αθροίσματος της αρχικής

εξίσωσης, μπορούν αυτοί οι όροι να βγουν έξω από αυτά τα αθροίσματα και να καταλήξουν στην εξίσωση:

$$L(\hat{u}_k) = L_{cyk,s} + L(u_k) + \log \frac{\sum_{\substack{(s',s) \\ u_k = +1}} \alpha_{k-1}(s') \beta_k(s) \gamma_k^e(s',s)}{\sum_{\substack{(s',s) \\ u_k = -1}} \alpha_{k-1}(s') \beta_k(s) \gamma_k^e(s',s)} = \\ = L_{cyk,s} + L(u_k) + L_e(u_k)$$

Το $L_{cyk,s}$ είναι η μέτρηση που πραγματοποίησε το κανάλι άρα αποτελεί την τιμή του καναλιού, το $L(u_k)$ αποτελεί την εκ των προτέρων πληροφορία και αντιπροσωπεύει την a priori πιθανότητα για το u_k και το $L_e(u_k)$ αποτελεί την extrinsic πληροφορία που θα πάρει από την διαδικασία της αποκωδικοποίησης όπου στον επόμενο αποκωδικοποιητή θα αποτελέσει την a priori πληροφορία.

Παρακάτω απεικονίζεται σε block διάγραμμα η διαδικασία της αποκωδικοποίησης ενός turbo κώδικα.

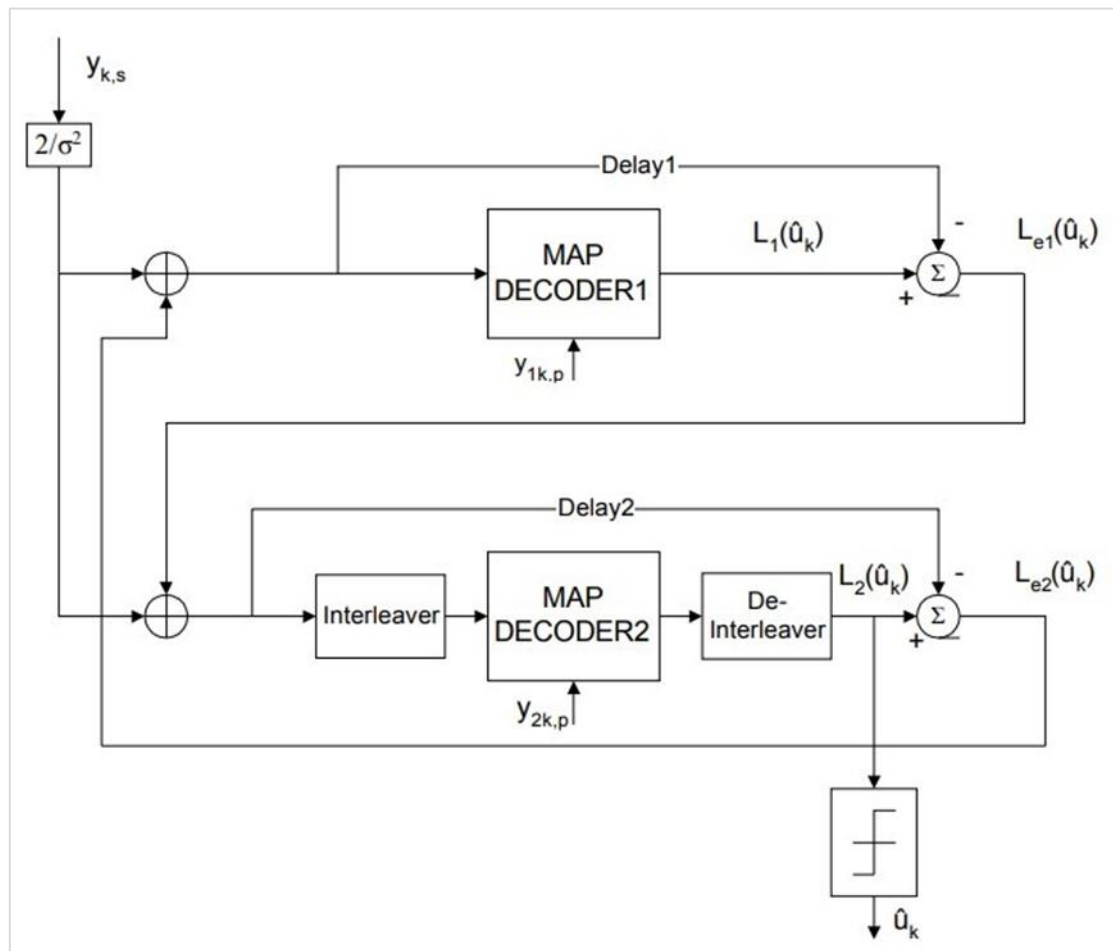


Σχήμα 19: Block διάγραμμα, διαδικασία της αποκωδικοποίησης ενός turbo κώδικα. Πηγή: Ιδία Επεξεργασία

Αποτελείται από δύο επιμέρους αποκωδικοποιητές MAP συνδεδεμένους παράλληλα (MAP DECODER1, MAP DECODER2). Η διαδικασία που ακολουθεί ο πρώτος αποκωδικοποιητής είναι να δέχεται ότι στέλνει ο πρώτος κωδικοποιητής, δηλαδή, ο $y_{1k,p}$ ο οποίος θα στείλει τα συστηματικά bit και τα bit ισοτιμίας. Από τη στιγμή όμως που ο MAP αλγόριθμος θα δώσει την τιμή $L(\hat{u}_k)$, για να πάρει την extrinsic πληροφορία θα χρησιμοποιήσει τον MAP DECODER2 και θα αφαιρέσει τις τιμές $L_{cyk,s}, L(u_k)$ από το $L(\hat{u}_k)$ και θα καταλήξει στην πράξη $L_{e1}(u_k) = L_1(\hat{u}_k) - L_{cyk,s} - L(u_k) = L_1(\hat{u}_k) - L_{cyk,s}$, όπου στην αρχή το $L(u_k) = 0$.

Για τα u_k η ακολουθία των soft τιμών θα χρησιμοποιηθεί ως a priori τιμή στον MAP DECODER2 και μαζί της θα εισαχθούν και τα συστηματικά bit. Πριν όμως από τον MAP DECODER2 θα περάσουν μέσα από έναν interleaver, για να διαταχθούν σύμφωνα με την ακολουθία των bit που θα περάσουν από τον $y_{2k,p}$, δηλαδή τον δεύτερο κωδικοποιητή. Αυτό στον turbo αποκωδικοποιητή αποτελεί την τρίτη είσοδο του MAP DECODER2 όπου στη συνέχεια θα παράγει την extrinsic πληροφορία κατά αυτόν τον τρόπο: $L_{e2}(u_k) = L_2(\hat{u}_k) - L_{cyk,s} - L(u_k)$. Στη συνέχεια, θα περάσει στην αντίστροφη διαδικασία του interleaver, στον de-interleaver, που θα χρησιμοποιηθεί ως a priori πληροφορία στον MAP DECODER1 μέχρι να πετύχει καλή απόδοση. Από εκείνη την στιγμή και έπειτα την τελευταία επανάληψη, ο MAP DECODER2 θα αποφασίσει για το σύμβολο που στάλθηκε σύμφωνα με τη σχέση $L(\hat{u}_k) = L_{cyk,s} + L_{e1}(u_k) + L_{e2}(u_k)$. Το $L(\hat{u}_k)$ θα είναι η αξιοπιστία της απόδοσης αλλά και το τελικό σύμβολο που στάλθηκε.

Παρακάτω απεικονίζεται το σχήμα με την αναλυτική δομή ενός turbo αποκωδικοποιητή:



Σχήμα 20: Αναλυτική δομή ενός turbo αποκωδικοποιητή. Πηγή: Ιδία Επεξεργασία

Ο αποκωδικοποιητής μέσω της extrinsic πληροφορία θα σημειώσει την αξιοπιστία της εκτίμησης του για την συνέχεια που θα ακολουθήσει βασισμένος στα bit ισοτιμίας που έλαβε χωρίς να περιλαμβάνει την λήψη των ακολουθιών των συστηματικών bits. Αφαιρείται λοιπόν, ο όρος $L_{cyk,s}$ και μαζί με αυτόν αφαιρείται και η πληροφορία που υπάρχει. Για αυτό το λόγο, η extrinsic πληροφορία θα εξαρτηθεί μόνο από τα bits ισοτιμίας της συγκεκριμένης χρονικής στιγμής και από όλα τα bits των προηγούμενων και επόμενων χρονικών στιγμών.

3.3.3 INTERLEAVER Σχεδίαση

Σημαντικό ρόλο για την απόδοση ενός Turbo κώδικα παίζει τόσο το είδος του interleaver όσο και το μέγεθος του. Ο interleaver είναι μία συσκευή που επηρεάζει την κατανομή των κωδικών λέξεων, διότι θα δεχτεί στην είσοδο κάποια σύμβολα και θα πάρει τα ίδια ακριβώς σύμβολα και στην έξοδο αλλά με διαφορετική σειρά. Στην ουσία, γράφει και διαβάζει τα σύμβολα με διαφορετικό τρόπο.

Παρακάτω φαίνεται ένα παράδειγμα ενός interleaver:

Πίνακας 1: Παράδειγμα ενός interleaver. Πηγή: Ιδία Επεξεργασία

X_1	X_2	X_3
X_4	X_5	X_6
X_7	X_8	X_9
X_{10}	X_{11}	X_{12}
X_{13}	X_{14}	X_{15}

Ο interleaver έχει τα bit εισόδου X_i που θα τα τοποθετήσει κατά σειρές αλλά θα τα διαβάσει κατά στήλες. Όταν θα φτάσει προς την έξοδο θα τα διαβάσει σε στήλες με τον εξής τρόπο: $X_1, X_4, X_7, X_{10}, X_{13}, X_2, X_5, X_8, X_{11}, X_{14}, X_3, \dots$

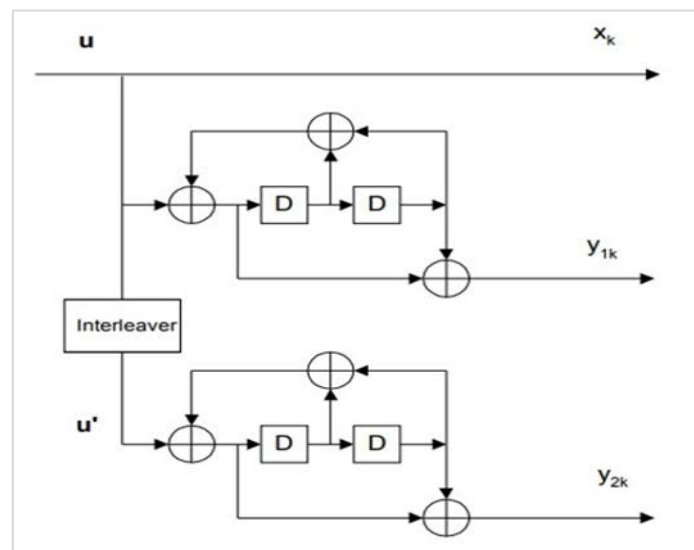
Υπάρχει όμως και ένα άλλο είδος interleaver που ονομάζεται ελικοειδής (helical interleaver) και διαβάζει τα σύμβολα με τον εξής τρόπο:

$X_{13}, X_{11}, X_9, X_4, X_2, X_{15}, X_{10}, X_8, X_6, X_1, X_{14}, \dots$

Ένα πρόβλημα που δημιουργείται εξαιτίας ενός interleaver είναι ότι δυσκολεύεται να τερματίσει ταυτόχρονα με τους επιμέρους RSC κωδικοποιητές που αποτελούν τον Turbo κωδικοποιητή, γιατί κάθε RSC θα δεχτεί διαφορετική είσοδο εξαιτίας της αναδιάταξης που έχει προκύψει από τον interleaver. Αν λοιπόν, προστεθούν κάποια bit στην είσοδο για να τερματίσει ο πρώτος κωδικοποιητής, το πιο πιθανό είναι ο δεύτερος να μην τερματίσει και το αντίστροφο. Ύστερα από πολλές έρευνες και θεωρίες η καλύτερη λύση που βρέθηκε για το πρόβλημα ήταν να κατασκευαστεί ένας interleaver που μετά την κωδικοποίηση όλων των bits

πληροφορίας, οι καταστάσεις των δύο επιμέρους RSC θα είναι ίδιες ώστε τα bit που θα τρέξουν εκείνη τη στιγμή να τερματιστούν ταυτόχρονα. Αξίζει να σημειωθεί ότι, τα bits που τρέχουν εκείνη τη στιγμή δε χρειάζεται να υποβληθούν σε interleaving αλλά να προστεθούν απλά στο τέλος της εισόδου, εφόσον, ολοκληρωθεί η διαδικασία του interleaving. Όσον αφορά το μέγεθος του τώρα, αυτό που συμβαίνει είναι ότι όσο μεγαλύτερο είναι τόσο καλύτερη απόδοση θα έχει ο κώδικας και σημαντικό ρόλο σε αυτή την περίπτωση παίζει ο interleaver για το πως θα διασκορπίσει τα bits μεταξύ τους και θα τα κάνει περισσότερο ασυσχέιστα. Στην αποκωδικοποίηση, κάθε MAP αποκωδικοποιητής θα δίνει σε κάθε bit μια διαφορετική εκτίμηση και θα την παρέχει ως μια extrinsic πληροφορία. Όσον αφορά το είδος του τώρα, αυτό που παίζει μεγαλύτερο ρόλο είναι ο βαθμός της τυχαιότητας με την οποία διασκορπίζει μία πιθανότητα σφάλματος από έναν κωδικοποιητή στον άλλο.

Παρακάτω είναι ένα παράδειγμα ενός Turbo κώδικα που αποτελείται από δύο επιμέρους RSC κώδικες των οποίων τα πολυώνυμα του θα είναι $(1, (101)_2/(111)_2)$.

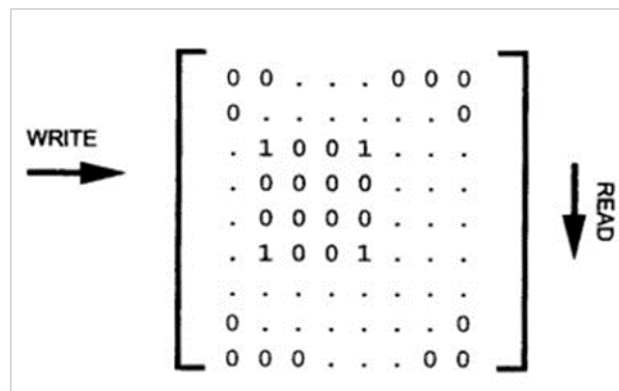


Σχήμα 21: Παράδειγμα ενός Turbo κώδικα. Πηγή: Ιδία Επεξεργασία

Η κωδική λέξη με ελάχιστο βάρος ($d_{\min}$) θα προκύψει από μία ακολουθία εισόδου. Αν οι δύο κωδικοποιητές είχαν αυτή την ακολουθία εισόδου $(00...01110...00)$ τότε θα προέκυπτε η χειρότερη περίπτωση, διότι το αποτέλεσμα θα ήταν μία κωδική λέξη $d_{\min}$. Όμως, η συγκεκριμένη περίπτωση είναι σχεδόν απίθανο να συμβεί. Έστω λοιπόν, μία άλλη ακολουθία η $(00...010010..00)$ που προκαλεί κωδικές λέξεις μικρού βάρους, θα οδηγήσει τον επιμέρους κωδικοποιητή σε απευθείας τερματισμό χωρίς κάποια bit που θα έτρεχαν στην είσοδο. Όμως, αν η συγκεκριμένη ακολουθία $(00..010010..00)$ θα αποτελέσει την είσοδο και για τους δύο επιμέρους κωδικοποιητές αυτό που θα προκύψει στην έξοδο θα είναι οι ακολουθίες

(00..010010..00), (00..01111..00) και (00..011110..00) με βάρος 10. Για να αποφύγει τις λέξεις μικρού βάρους ο interleaver, πρέπει να διασπάσει την ακολουθία 010010.

Συμπερασματικά λοιπόν, η απόδοση ενός Turbo κώδικα εξαρτάται από τους συνδυασμούς που θα κάνει ο interleaver στις ακολουθίες εισόδου που προκαλούν κωδικές λέξεις με μικρό βάρος, με ακολουθίες που αποδίδουν μεγαλύτερα βάρη ώστε η κωδική λέξη να είναι μεγάλη. Τα καλύτερα αποτελέσματα προκύπτουν από random interleaver. Όσο αυξάνεται το βάρος τόσο μειώνεται η πιθανότητα να συμβεί ένας συνδυασμός λέξεων μικρού βάρους. Με την χρήση δύο επιμέρων RSC κωδικοποιητών η πιθανότητα να εμφανιστεί μικρό βάρος μειώνεται, αλλά με διαφορετικούς interleavers. Όταν όμως προκύπτει ένας άλλος συνδυασμός, δηλαδή αυτός της random interleaver θα προκύψουν πάλι τα ίδια αποτελέσματα αλλά πρέπει ξανά να ικανοποιηθούν κάποια κριτήρια για την κατασκευή τους και συγκεκριμένα στην περίπτωση με βάρος, που θα προκαλέσει δύσκολη περίπτωση. Στο παρακάτω παράδειγμα, απεικονίζεται ο interleaver που αποτυγχάνει να διασπάσει την αρχική είσοδο αφού και μετά το interleaving θα είναι πάλι ίδια. Αποτελείται από μία ακολουθία εισόδου με βάρος 4 και κάθε φορά επιβάλλει να γίνει λεπτομερής μελέτη του interleaving σε σύγκριση με όλες τις πιθανές εισόδους.



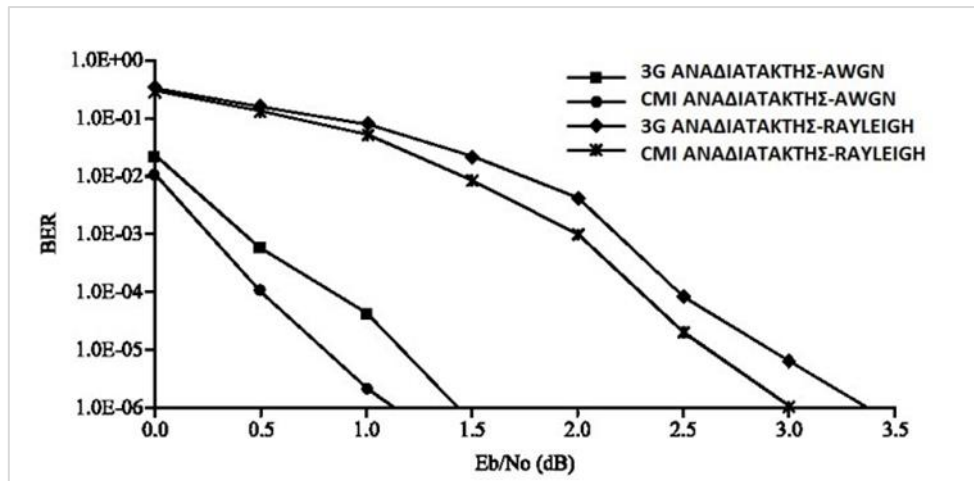
3.3.4 TURBO Κώδικες σε Συστήματα 3G

Το 3G αποτελεί την Τρίτη γενιά ασύρματων επικοινωνιών. Είναι η αναβάθμιση του δικτύου 2G, 2,5G, GPRS και 2,7G EDGE το οποίο προσφέρει πιο γρήγορη μεταφορά δεδομένων και καλύτερη ποιότητα φωνής. Ο όρος 3G καλύπτει ένα τεχνολογικό φάσμα ασύρματων δικτύων που συμπεριλαμβάνει το CDMA-2000, WCDMA, UMTS και EDGE. Το UMTS είναι η τεχνολογία μετάδοσης ομιλίας και δεδομένων υψηλής ταχύτητας που αποτελεί μέρος της 3ης γενιάς των ασύρματων προτύπων. Το WCDMA είναι η τεχνολογία που χρησιμοποιείται στο UMTS και συνδέεται άμεσα με το 3G, σαν πρότυπο διασύνδεσης τηλεπικοινωνιών και δεδομένων. Το EDGE είναι το πρότυπο επικοινωνιών που έδινε στο σύστημα επικοινωνιών GSM, την ευκαιρία να ανταπεξέλθει στο δίκτυο 3G, όσον αφορά τον τομέα ταχύτητας

μετάδοσης δεδομένων. Παρόλα αυτά όμως, δεν κατάφερε να ανταγωνιστεί στον τομέα της επικοινωνίας και συχνά αναφέρεται ως 2,75G.

Σε ένα σύστημα ασύρματης επικοινωνίας 3G παρουσιάστηκε ένα εξελιγμένο σύστημα κωδικοποίησης Turbo, το CMI(Code Matched Interleavers). Είναι παρεμβολείς-αναδιατάκτες προσαρμοσμένοι πάνω στα χαρακτηριστικά των συνελκτικών κωδικοποιητών. Σε έναν turbo κώδικα, οι παρεμβολείς-αναδιατάκτες βρίσκονται πάνω σε δύο συστήματα 3G, το WCDMA και το CDMA-2000. Σημαντικό παράγοντα όμως αποτελεί η σωστή σχεδίαση τους πάνω στο βάρος του συστήματος που θα πρέπει να μελετηθεί σωστά. Οι κώδικες αυτοί ήρθαν στην επιφάνεια το 2002 και προτιμώνται στις επικοινωνίες με δίκτυο 3G συνδυάζοντας ψευδοτυχαίους και S-τυχαίους παρεμβολείς, με αποτέλεσμα να εξαφανίζουν ή να μειώνουν τις λέξεις χαμηλού βάρους του κώδικα. Σε S-τυχαίους παρεμβολείς, αυτό που συμβαίνει είναι ότι αρχικά πρέπει να διαχωρίσει τη θέση στοιχείων μέσα σε ένα παράθυρο όσο το δυνατόν περισσότερο μπορεί, έτσι ώστε κάθε δύο στοιχεία εντός του παραθύρου με μέγεθος S να μην εντοπίζεται σε παράθυρο μεγέθους S την ίδια χρονική στιγμή. Ο ψευδοτυχαίος σχεδιασμός αναδιατάκτη είναι να αφαιρέσει κάποια φαινομενική σειρά έτσι ώστε οι ακολουθίες εξόδου ισοτιμίας χαμηλού βάρους να μην συμβαίνουν και στους δύο πάνω-κάτω RSC κωδικοποιητές την ίδια χρονική στιγμή. Αυτό που θέλει να πετύχει ουσιαστικά, είναι να γίνει εξάλειψη των λέξεων του κώδικα χαμηλού βάρους με αποτέλεσμα την βελτίωση των ποσοστών σφαλμάτων ή τη μείωση των λέξεων χαμηλού βάρους του κώδικα οι οποίες δεν μπορούν να εξαλειφθούν. Η εξάλειψη αυτή μπορεί να γίνει με ένα διαχωρισμό της εισόδου που δημιουργεί τη συγκεκριμένη λέξη κώδικα. Ο αναδιατάκτης με S-τυχαίους περιορισμούς μπορεί είτε να διακόψει μικρό βάρος με ακολουθίες 3 εισόδων με μήκη έως S+1 είτε να το επεκτείνει σε μεγαλύτερο βάρος πάνω από 2, όπου $S < 2$. Για την επιβεβαίωση καταλληλότητας του CMI στα δύο συστήματα επικοινωνίας 3G πρέπει να γίνει μια προσομοίωση(FEC-FORWARD ERROR CORRECTION) της απόδοσης τους.

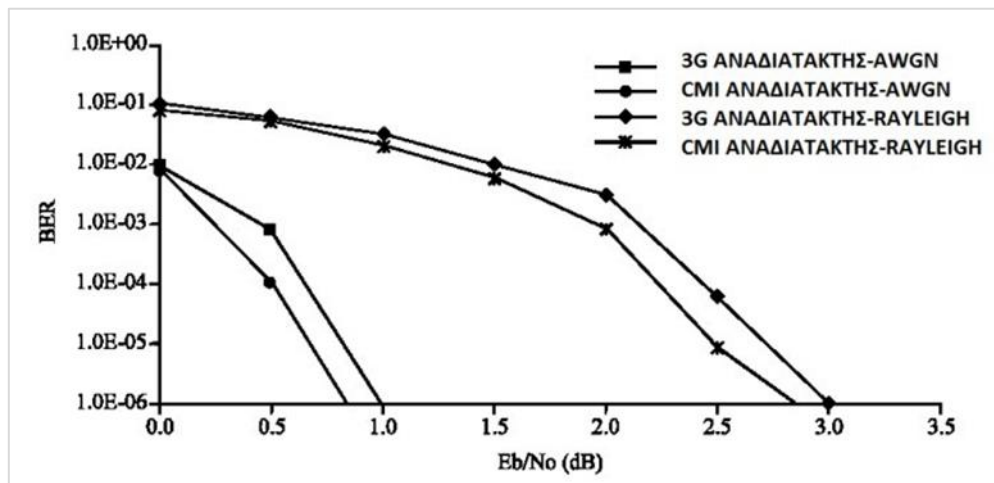
Αποτέλεσμα προσομοίωσης (FEC) για το σύστημα WCDMA:



Σχήμα 22: Αποτέλεσμα προσομοίωσης (FEC) για το σύστημα WCDMA. Πηγή: <https://www.researchgate.net/>

Για το σύστημα WCDMA, οι προσομοιώσεις πραγματοποιήθηκαν για μήκος πληροφοριών $N=5114$, με ρυθμό κώδικα $R=1/3$ και με αποκωδικοποιητή Log-MAP με μέγιστο αριθμό επαναλήψεων αποκωδικοποίησης 8 και 14 επί θορύβου AWGN, με ασυσχέτιστο κανάλι εξασθένισης Rayleigh. Αυτό που παρατηρείται είναι ότι οι CMI αναδιατάκτες αποδίδουν καλύτερα από τους 3G αναδιατάκτες, με κέρδος κωδικοποίησης 0,2-0,35dB.

Αποτέλεσμα προσομοίωσης (FEC) για το σύστημα CDMA-2000:



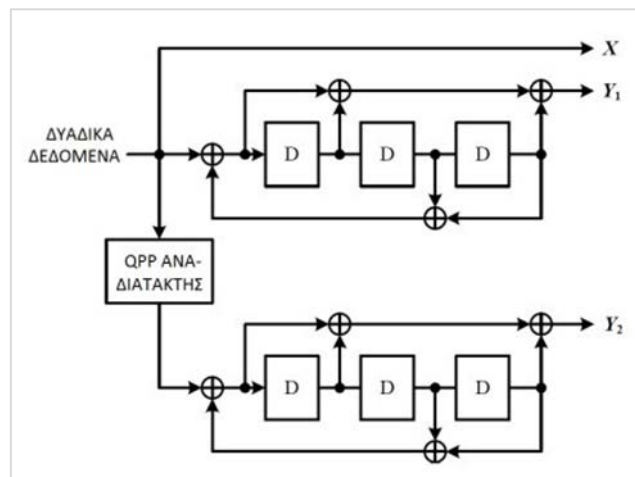
Σχήμα 23: Αποτέλεσμα προσομοίωσης (FEC) για το σύστημα CDMA-2000. Πηγή: <https://www.researchgate.net/>

Για το σύστημα CDMA-2000, οι προσομοιώσεις πραγματοποιήθηκαν για μήκος πληροφοριών $N=6138$, με ρυθμό κώδικα $R=1/3$ και με αποκωδικοποιητή Log-MAP με μέγιστο αριθμό επαναλήψεων αποκωδικοποίησης 9 για το κανάλι AWGN, με ασυσχέτιστο κανάλι εξασθένισης Rayleigh. Για το κανάλι AWGN το πλάτος εξασθένισης είναι $\alpha=1$ και για το κανάλι Rayleigh είναι 0,8662. Αυτό που παρατηρείται είναι ότι οι CMI αναδιατάκτες αποδίδουν καλύτερα από τους 3G αναδιατάκτες, με κέρδος κωδικοποίησης 0,15-0,3dB.

3.3.5 TURBO Κώδικες σε Συστήματα 4G

Το 4G αποτελεί την 4η γενιά της τεχνολογίας κυψελοειδών δικτύων, που διαδέχθηκε το 3G. Ένα σύστημα 4G πρέπει να παρέχει δυνατότητες που ορίζονται από την ITU στο IMT advanced. Ένα υποψήφιο σύστημα πρώτης έκδοσης του 4G αποτελεί το LTE (Long Term Evolution) το οποίο έχει αναπτυχθεί και εμπορικά. Στους Turbo κώδικες τώρα, το 4G χρησιμοποιεί συνενωμένους κώδικες, οι οποίοι είναι πιο εξελιγμένοι στη διόρθωση σφαλμάτων και στη μετάδοση των δεδομένων με υψηλές ταχύτητες. Χρησιμοποιεί Log-MAP και MAX-Log-MAP αλγόριθμους, οι οποίοι μειώνουν την αριθμητική πολυπλοκότητα αλλά ταυτόχρονα μπορούν να διατηρήσουν σε καλό στάδιο τα επίπεδα αποκωδικοποίησης. Μεγάλη χρήση στις Τηλεπικοινωνίες 4G αποτελούν οι Δυαδικοί κώδικες (LTE) και οι Διπλοί δυαδικοί κώδικες.

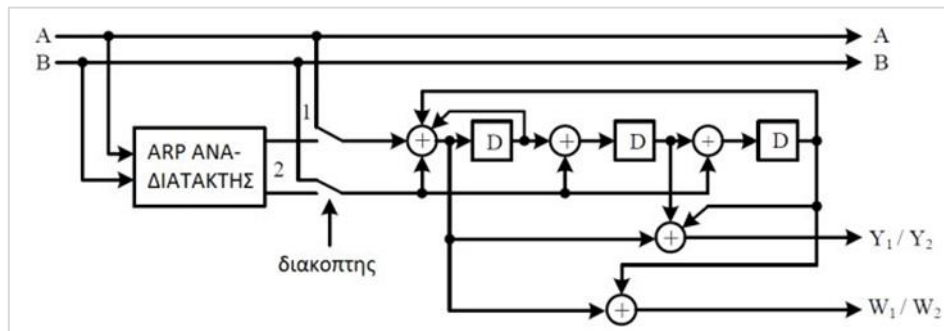
Παρακάτω απεικονίζεται ένας Δυαδικός κώδικας Turbo:



Σχήμα 24: Δυαδικός κώδικας Turbo. Πηγή: Ιδία Επεξεργασία

Ο συγκεκριμένος Δυαδικός κώδικας Turbo βρίσκεται στο πρότυπο 3GPP (LTE), ο οποίος είναι ένας παράλληλος συνενωμένος συνελκτικός κώδικας με δύο οκταδικούς κωδικοποιητές και έναν αντιμεταθετικό αναδιατάκτη 2ου βαθμού. Ο ρυθμός του κώδικα είναι $R=1/3$ και αποτελείται από δύο συνελκτικούς κωδικοποιητές που συνδέονται με έναν αναδιατάκτη. Άρα, ο N αριθμός των δεδομένων κωδικοποιείται σε $3N$ bits δεδομένων. Η συνάρτηση μεταφοράς είναι: $G(D) = [1, g_1(D)/g_0(D)]$, με $g_0(D) = 1+D^2+D^3$ και $g_1(D) = 1+D+D^3$. Η τιμή που θα έχουν οι καταχωρητές μετατόπισης είναι όλα μηδενικά ώστε να κωδικοποιήσουν δυαδικά ψηφία εισόδου. Ο τερματισμός θα γίνει με την λήψη των bits που θα τρέχουν εκείνη την στιγμή από την ανατροφοδότηση του καταχωρητή μετατόπισης, αφού πρώτα κωδικοποιηθούν όλα. Η δουλειά του αναδιατάκτη σε αυτή την περίπτωση είναι να λάβει κάθε N bits που εισέρχεται και να το μπερδεύει με έναν ψευδοτυχαίο τρόπο.

Παρακάτω απεικονίζεται ένας Διπλός Δυναδικός κώδικας Turbo:



Σχήμα 25: Δυναδικός κώδικας Turbo. Πηγή: Ιδία Επεξεργασία

Ο Διπλός Δυναδικός κώδικας Turbo βρίσκεται στο πρότυπο Wimax-IEEE 802.16e όπου χρησιμοποιεί έναν διπλό δυναδικό κυκλικό αναδρομικό συστηματικό συνελκτικό κώδικα. Υπάρχουν δύο ζευγάρια δεδομένων A και B όπου η τροφοδότηση τους γίνεται στο κυκλικό αναδρομικό σύστημα συστηματικού συνελκτικού κώδικα αλλά δύο φορές. Επίσης, υπάρχουν 4 bits ισοτιμίας τα (Y1,W1) και (Y2,W2) που λειτουργούν σε φυσιολογική και παρεμβλλόμενη ροή. Τα σύμβολα βρίσκονται ως εξής:

Ανατροφοδότηση: $1+D+D^3$

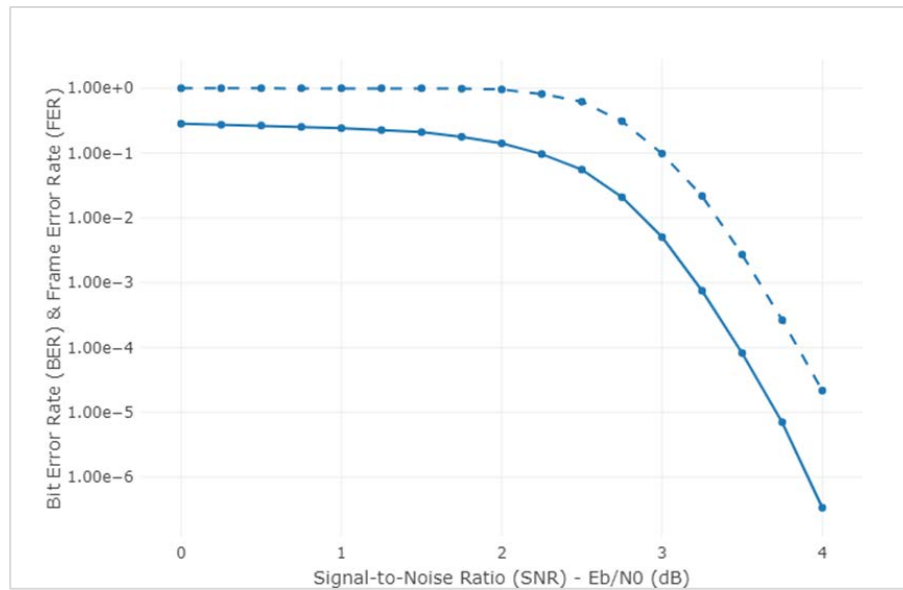
Y: $1+D^2+D^3$

W: $1+D^3$

Ο τερματισμός σε αυτή την περίπτωση ισούται με την τελική κατάσταση των χωροδιχτυωμάτων και της τελικής του κατάστασης. Η εκτέλεση αυτή πρέπει να γίνει πριν την κωδικοποίηση ώστε να καθοριστεί η αρχική του κατάσταση.

3.3.6 AFF3CT Προσομοίωση

Για την καλύτερη μελέτη ενός Turbo κώδικα στο παρακάτω σχήμα απεικονίζεται ένα παράδειγμα προσομοίωσης και μεταβολής της απόδοσης του πάνω στον προσομοιωτή AFF3CT.



Σχήμα 26: συστηματικός Turbo κώδικας (3084,1024). Πηγή: <https://imtech.imt.fr>

Στο σχήμα απεικονίζεται ένας συστηματικός Turbo κώδικας (3084,1024). Ο ρυθμός του κώδικα είναι $R=1/3$, ο αριθμός των bits που στέλνονται είναι $K=1024$, ενώ το κανάλι είναι AWGN. Στην ουσία ο αριθμός των bits αποτελεί το μέγεθος του interleaver. Ο interleaver στη συγκεκριμένη περίπτωση είναι ο LTE. Σκοπός ενός interleaver, όπως έχει αναφερθεί, είναι να μπερδεύει την ακολουθία εισόδου έτσι ώστε αν δημιουργηθούν λάθη στα συνεχόμενα bit μετά το interleaving να εμφανιστούν ως μεμονωμένα σφάλματα για να γίνει η διόρθωσή τους πιο εύκολη. Ο interleaver εξαρτάται και από το μέγεθος του και αυτό που ισχύει είναι ότι όσο μεγαλύτερος είναι τόσο πιο μπερδεμένα θα είναι τα bits της ακολουθίας εισόδου. Στη συνέχεια σημαντικό παράγοντα αποτελεί ο αριθμός των επαναλήψεων στο παραπάνω σχήμα $i=6$ που θα εκτελεστεί η αποκωδικοποίηση που γίνεται με αλγόριθμο MAP(BCJR). Γενικά, όσες περισσότερες επαναλήψεις πραγματοποιηθούν τόσο καλύτερα αποτελέσματα θα υπάρξουν. Αυτό συμβαίνει γιατί το σύστημα θα διορθώσει περισσότερα λάθη που θα γίνουν κατά την μετάδοση με αποτέλεσμα να μειωθεί η πιθανότητα σφάλματος. Επίσης, όσο αυξάνεται ο αριθμός των συμβόλων τόσο καλύτερη(μικρότερη) πιθανότητα σφάλματος θα πάρει. Το 3084 είναι το μέγεθος πλαισίου(N_{cw}) και το μέγεθος της κωδικής λέξης (N). Το BER τώρα, συμβολίζει το ποσοστό των bit που έχουν γίνει τα σφάλματα σε σχέση με το συνολικό αριθμό των bit που πραγματοποιήθηκαν κατά την μετάδοση και το FEC συμβολίζει το ποσοστό σφάλματος πλαισίου, δηλαδή, ο αριθμός των πακέτων που μεταδίδονται μαζί με τα σφάλματα προς το συνολικό αριθμό πακέτων δεδομένων που στέλνονται. Συμπερασματικά λοιπόν, αυτό που φαίνεται από το σχήμα είναι ότι υπάρχει μία καμπύλη προς τα κάτω με τις ανάλογες τιμές

και ουσιαστικά ο κώδικας θα τερματίσει με τις αποτελέσματα που δείχνει η παραπάνω γραφική παράσταση.

3.4 POLAR Κώδικας

Σύμφωνα με τις ιστοσελίδες <http://www.alexiosbalatsoukas.com/>, <https://www.computerworld.com/> και <https://www.retailnews.asia/> δύνανται να ειπωθούν οι παρακάτω αναφορές σχετικά με τον κώδικα Polar.

Το 2008 ο Erdal Arıkan εισήγαγε τους πολικούς κώδικες. Μία νέα μέθοδο κωδικοποίησης γραμμικού κώδικα διόρθωσης σφαλμάτων, η οποία υπερτερεί των συνελκτικών κωδίκων LDPC σε απόδοση διόρθωσης σφαλμάτων και ρυθμού μετάδοσης. Η επινόηση των Polar Codes θεωρείται ένα από τα μεγαλύτερα επιτεύγματα της εποχής καθώς έχουν άμεση εφαρμογή σε τεχνικές κωδικοποίησης καναλιού σε δίκτυα 5G (5ης γενιάς). Σημαντικό για κάθε κώδικα είναι να επιτευχθεί η σωστή κωδικοποίηση της πληροφορίας με χαμηλή πολυπλοκότητα. Η πολυπλοκότητα καθορίζει την μνήμη που χρειάζεται, την ισχύ που καταναλώνει κ.α. Συγκεκριμένα, η πολυπλοκότητα κωδικοποίησης και αποκωδικοποίησης των πολικών κωδίκων είναι μέτρια και αυτό τους καθιστά πολύ χρήσιμους.

Το 2016 η κινεζική εταιρεία τηλεπικοινωνιακών δικτύων και υπηρεσιών Huawei ανακοινώνει την επίτευξη 27 Gbps στην κάτω ζεύξη (downlink) σε δίκτυα 5G. Ο πολικός κώδικας είχε τριπλάσια απόδοση φάσματος σε σχέση με τα ισχύοντα δίκτυα RAN. Έπειτα, η Huawei σε συνεργασία με την канаδική εταιρία τηλεπικοινωνιών Telus γνωστοποιούν τα αποτελέσματα δοκιμών 5G φτάνοντας τις μέγιστες ταχύτητες των 29,3 Gbps, συγκριτικά 200 φορές πιο γρήγορες από το υπάρχον πρότυπο LTE (Long Term Evolution). Τα πειράματα πραγματοποιήθηκαν σε πραγματικές συνθήκες και σε σπίτια των εργαζομένων της Telus στο Βανκούβερ.

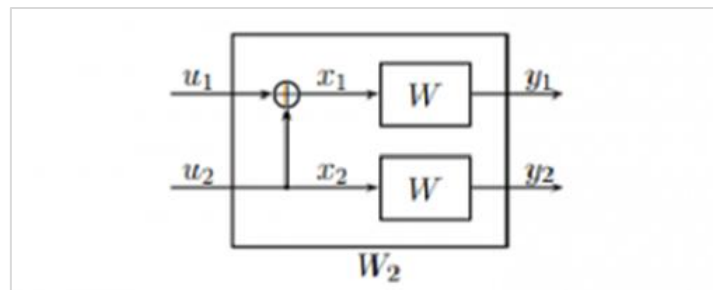
Οι κώδικες Polar εντάσσονται στους γραμμικούς μπλοκ κώδικες (linear block codes), αποτελούνται από κωδικές λέξεις οι οποίες προέρχονται από την εφαρμογή γραμμικών πράξεων στο αρχικό διάνυσμα της πληροφορίας. Η κατασκευή των πολικών κωδίκων βασίζεται σε μία πολλαπλή αναδρομική εφαρμογή ενός αρχικού φυσικού καναλιού επικοινωνίας χωρίς μνήμη σε έναν αριθμό υποκαναλιών bit. Όταν ο αριθμός των αναδρομών είναι υψηλός, τότε τα υποκανάλια bit τείνουν να έχουν είτε μεγάλη αξιοπιστία είτε χαμηλή (αθόρυβα ή καθαρά θορυβώδη κανάλια αντίστοιχα). Δηλαδή με άλλα λόγια να πολώνονται ή να γίνονται αραιά, με τα bits των δεδομένων να μοιράζονται στα πιο αξιόπιστα κανάλια (στα αθόρυβα). Ο Arıkan απέδειξε το φαινόμενο της πόλωσης των πολικών κωδίκων και συνεπώς

το γεγονός ότι επιτυγχάνουν χωρητικότητα καναλιού. Αξιοποιώντας τα παραπάνω, υπάρχει η δυνατότητα επιλογής, δηλαδή, τέλεια κανάλια για την μετάδοση των δεδομένων. Για να παγώσουν τα κανάλια τώρα, που είναι άχρηστα μέσω μίας σταθερής τιμής που ο πομπός και ο δέκτης, θεωρούν τη μετάδοση τιμής η οποία είναι γνωστή (σταθερή) μέσω των καναλιών άρα μειώνεται η πιθανότητα λάθους. Αυτός είναι ο τρόπος για την δημιουργία κωδικών με μειωμένο ποσοστό σφάλματος μετάδοσης (error rate).

3.4.1 Τεχνική Πόλωσης Καναλιού

Η πόλωση καναλιού είναι μία διαδικασία κατά την οποία παράγονται N κανάλια από N ανεξάρτητα αντίγραφα ενός δοσμένου B-DMC W με χωρητικότητα $C(W)$, η οποία θα είναι κοντά στο 0 (άχρηστο κανάλι) ή κοντά στο 1 (χρήσιμο κανάλι). Η πόλωση του καναλιού αποτελείται από δύο στάδια, σύμφωνα με τον Arika, η πρώτη φάση ονομάζεται συνδυασμός καναλιών (channel combining) και η δεύτερη διαχωρισμός καναλιών (channel splitting).

1) Συνδυασμός καναλιών (Channel combining) : Είναι μία διαδικασία όπου συνδυάζονται αναδρομικά N αντίγραφα των δεδομένων B-DMC W , έτσι ώστε να δημιουργηθεί το διάνυμα καναλιού $W(N) : X^N \rightarrow Y^N$, όπου $N=2n$, $n \leq 0$. Ο μετασχηματισμός G_n θα πρέπει να είναι όπως στην παρακάτω εικόνα, δηλαδή χαμηλής πολυπλοκότητας και υλοποιημένος ως εξής :



Σχήμα 27: Συνδυασμός καναλιών. Πηγή: Ιδία Επεξεργασία

Σύμφωνα με την εικόνα τα δύο κανάλια W συνδυάζονται και δημιουργείται το καινούριο κανάλι W_2 αφού ο γραμμικός μετασχηματισμός G_n μεταξύ των u_1, u_2 και x_1, x_2 έχουν αντιστοιχία 1-1. Οπότε σε αυτήν την περίπτωση έχουμε :

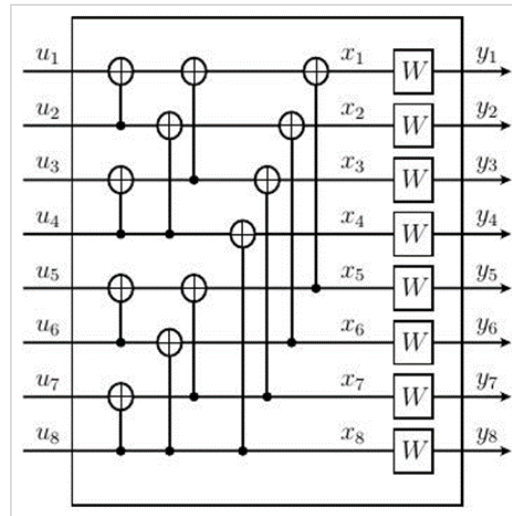
$$W_2(y_1, y_2 | u_1, u_2) = W(y_1 | u_1 \oplus u_2) W(y_2 | u_2)$$

$$I(u_1, u_2; y_1, y_2) = I(x_1, x_2; y_1, y_2) = 2I(W)$$

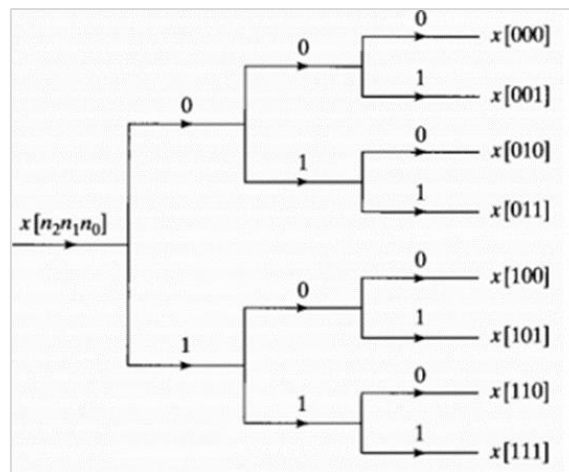
Για $N = 2n$ συνδυασμούς καναλιών γίνεται με τον εξής τρόπο :

$$W_N(Y^N | U^N) = W_{N/2}(Y^{N/2} | U^{N/2} \oplus U^{N/2}) W_{N/2}(Y^{N/2+1} | U^{N/2+1})$$

Παρακάτω, το παράδειγμα είναι για $N = 8$ με 8 ανεξάρτητα κανάλια W έχοντας ως αποτέλεσμα το διάνυσμα $W8$:



Στην επόμενη εικόνα φαίνεται η αναδρομικότητα (bit-reversal) προκειμένου να δημιουργηθούν τα κανάλια διανυσμάτων :



2) Διαχωρισμός καναλιών (channel splitting) : Σε αυτό το στάδιο, το διάνυσμα WN που δημιουργήθηκε στην πρώτη φάση, χωρίζεται σε N κανάλια WN^i :

$$\{0,1\} \rightarrow Y^N \times \{0,1\}^{i-1}, \text{ όπου } 1 \leq i \leq N.$$

Σχετικά με την κοινή πληροφορία και σύμφωνα με τον κανόνα της αλυσίδας προκύπτει το εξής: $I(U1, U2; Y1, Y2) = I(U1; Y1, Y2) + I(U; Y1, Y2, U1)$

Από την σχέση παραπάνω παρατηρείτε, ότι το αριστερό μέλος είναι η κοινή πληροφορία που μοιράζονται οι όροι $U1$ και $Y1Y2$. Θεωρώντας ότι το κανάλι αυτό είναι το W^- : $\{0,1\}^2 \rightarrow Y2$. Έπειτα το δεύτερο μέρος του δεξιού μέλους είναι η κοινή πληροφορία μεταξύ των $U2$ και της

εξόδου εφόσον το U_1 μας είναι γνωστό. Υποθέτοντας ότι το συγκεκριμένο κανάλι είναι το W^+ . Σύμφωνα λοιπόν με τον Arikan, οι πιθανότητες μετάδοσης των δύο καναλιών είναι :

Το κανάλι W διαχωρίζεται σε δύο κανάλια W^- και W^+

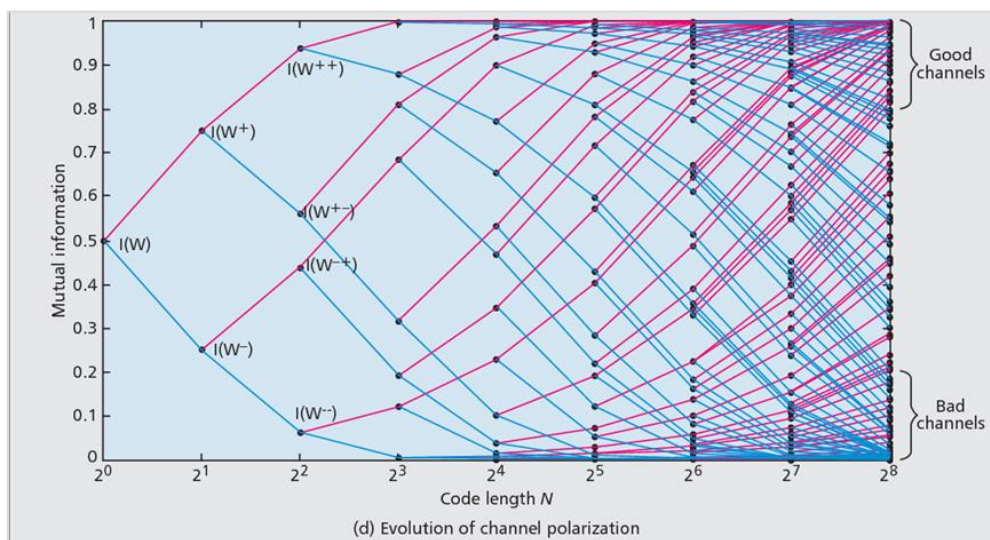
$$W^-(y_1, y_2 | u_1) = \frac{1}{2} \sum_{u_2 \in \{0,1\}} W(y_1 | u_1 \oplus u_2) W(y_2 | u_2)$$

$$W^+(y_1, y_2, u_1 | u_2) = \frac{1}{2} W(y_1 | u_1 \oplus u_2) W(y_2 | u_2)$$

Η παρακάτω ιδιότητα ισχύει για την κοινή πληροφορία που μοιράζονται τα δύο νέα κανάλια:

$$I(W^+) + I(W^-) = 2I(W), \text{ όπου } I(W^+) \geq I(W) \geq I(W^-)$$

Από την εικόνα παρατηρείτε, ότι το κανάλι W^+ τείνει προς το 1 και το αντίστοιχο W^- τείνει προς το 0, όμως το άθροισμά τους είναι $2I(W)$.



Σχήμα 28: Παράδειγμα πόλωσης. Πηγή: <https://www.computerworld.com/>

Παρακάτω φαίνεται σύμφωνα με τον Arikan η διαδικασία πόλωσης ενός καναλιού. Οι Polar Codes συνήθως δομούνται με μήκος δύναμης του 2 ($N = 2^n$). Η γεννήτρια G_n δημιουργείται με την n -οστή δύναμη του Kronecker του πίνακα:

$$F_2 = \begin{bmatrix} 1 & 0 \\ 1 & 1 \end{bmatrix}$$

Όπου n , ο λογάριθμος με βάση το 2 του μεγέθους του block του κώδικα.

$$\text{Γενικά ισχύει ότι } F^{\otimes m} = F^{\otimes(m-1)} \otimes F = \begin{bmatrix} F^{\otimes(m-1)} & 0 \\ F^{\otimes(m-1)} & F^{\otimes(m-1)} \end{bmatrix}, \text{ με } F^{\otimes 0} = [1]$$

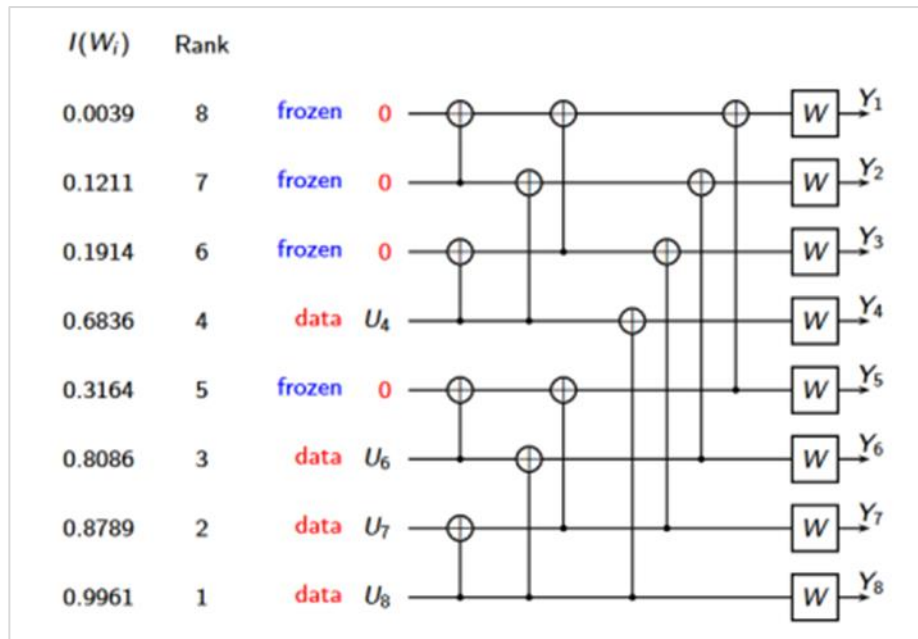
Επομένως ο $F^{\otimes 2}$ θα οριστεί ως εξής :

$$F^{\otimes 2} = \begin{bmatrix} 1 \cdot \begin{bmatrix} 1 & 0 \\ 1 & 1 \end{bmatrix} & 0 \cdot \begin{bmatrix} 1 & 0 \\ 1 & 1 \end{bmatrix} \\ 1 \cdot \begin{bmatrix} 1 & 0 \\ 1 & 1 \end{bmatrix} & 1 \cdot \begin{bmatrix} 1 & 0 \\ 1 & 1 \end{bmatrix} \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 \\ 1 & 1 & 1 & 1 \end{bmatrix}$$

Η παραπάνω διαδικασία συνεχίζεται επαναληπτικά και για μεγαλύτερα μεγέθη block. Όπου $\oplus$ είναι η δύναμη Kronecker σε κάθε περίπτωση. Με αυτόν τον τρόπο δημιουργείται το διάνυσμα καναλιών μήκους N , όπως αναφέρεται και παραπάνω, από τα οποία χρησιμοποιούνται μερικά εξ αυτών. Τα κατάλληλα κανάλια που επιλέγονται χαρακτηρίζονται ως bit πληροφορίας (information bits) και τα υπόλοιπα χαρακτηρίζονται ως bits σταθερής τιμής (frozen bits). Το διάνυσμα U αποτελείται από το διάνυσμα N συγκεκριμένων θέσεων. Τοποθετούνται λοιπόν σε αυτές τις θέσεις τα ιδανικά bit πληροφορίας (information bits) και παγώνουν οι υπόλοιπες θέσεις που ανήκουν στο σύνολο των bits σταθερών τιμών (frozen bits) με μια σταθερή τιμή, όπως 0.

3.4.2 Κωδικοποίηση κατά ARIKAN

Μετά την διαδικασία κατασκευής του καναλιού, που αναφέρθηκε παραπάνω, απαραίτητος είναι ο υπολογισμός της συμμετρικής χωρητικότητας του συνθετικού καναλιού που ορίζεται ως $I(W)$. Η διαδικασία που ακολουθεί έχει ως εξής : τα κανάλια με την μικρότερη συμμετρική χωρητικότητα δεν χρησιμοποιούνται για εκπομπή, δηλαδή η είσοδός τους «παγώνετε» και η τιμή τους έχει καθοριστεί εκ των προτέρων (συνήθως είναι 0). Ο ρυθμός του κώδικα που έχει επιλεγεί διαμορφώνει πόσα κανάλια πρέπει να παγώσουν. Τα υπόλοιπα κανάλια που απομένουν, θα χρησιμοποιηθούν για να μεταδοθεί καθαρή η πληροφορία χωρίς όμως να έχουν προστεθεί τα πλεονάζοντα bit πληροφορίας. Τα κανάλια αυτά θεωρούνται αξιόπιστα καθώς η χωρητικότητά τους είναι σχεδόν 1. Ποια θα είναι αυτά τα κανάλια εξαρτάται από την επιλογή του πίνακα γεννήτριας G_n . Οι γραμμές του πίνακα θα είναι εκείνες που θα χρησιμοποιηθούν για να γίνει η σύνθεση των καναλιών που θα μεταφέρουν την πληροφορία.

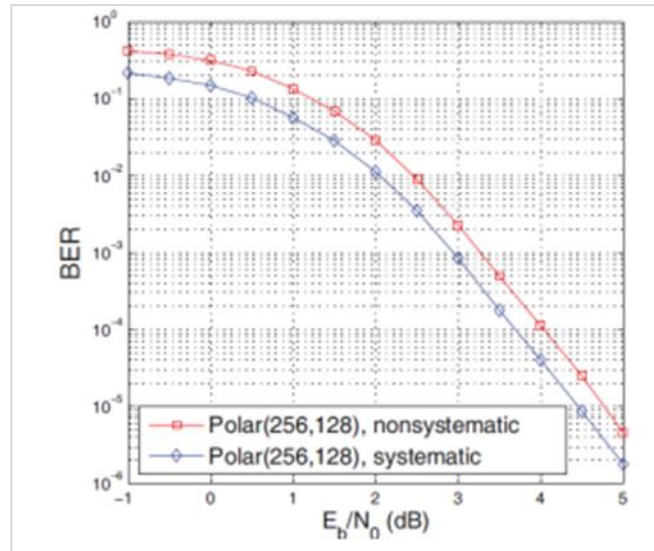


Σχήμα 29: Κωδικοποίηση Polar για Rank=8. Πηγή: <https://www.computerworld.com/>

Εδώ από τα αρχικά 8 bits τα 4 αποτελούν πληροφορία, άρα, ο ρυθμός του κώδικα είναι 1/2. Η κωδικοποίηση πραγματοποιείται σε $N \log N$ βήματα, όπου N το μέγεθος του μπλοκ (εδώ 8).

3.4.2.1 Μεταγενέστερες Προτάσεις Κωδικοποίησης

Η προσπάθεια για βελτίωση της μεθόδου της κωδικοποίησης, έπειτα από την πρόταση του Arikan, δεν άργησε να εξεταστεί από την ερευνητική κοινότητα. Η ανάγκη που δημιουργήθηκε για συνολική επίδοση ενός επικοινωνιακού συστήματος που κάνει χρήση του συγκεκριμένου αλγορίθμου, θα πραγματοποιούνταν διατηρώντας την πολυπλοκότητα την υλοποίησης με την βοήθεια άλλων μεθόδων ή με τρόπους που θα μείωναν την πολυπλοκότητα. Οι κώδικες σε συστηματική μορφή, όπως δηλαδή οι πολικοί, διαθέτουν πλεονεκτήματα κατά την διαδικασία της αποκωδικοποίησης εξαιτίας των χαρακτηριστικών τους. Η μετατροπή λοιπόν, των γραμμικών κωδικών σε συστηματική μορφή είναι μία σημαντική λύση διότι λόγω του διαχωρισμού πληροφορίας και πλεονασμού ελαττώνεται η διάδοση των σφαλμάτων. Ο ίδιος ο Arikan, σε άρθρο του κοινοποίησε τα αποτελέσματα αυτής της έρευνας. Απέδειξε ότι, η κωδικοποίηση των Polar κωδικών μπορεί να γίνει με την ίδια πολυπλοκότητα $O(N \log N)$ σε συστηματική μορφή.

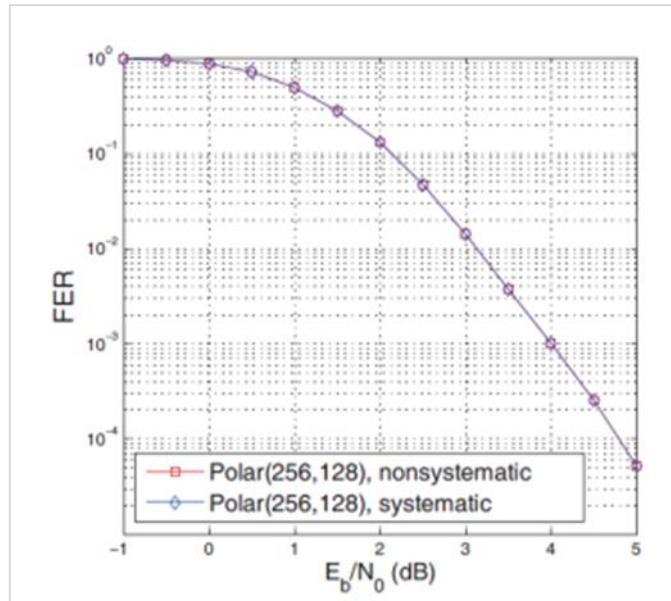


Σχήμα 30: Σύγκριση Κέρδους σε systematic, non-systematic. Πηγή: <https://www.computerworld.com/>

Υπάρχουν δύο τρόποι για αυτήν την μετατροπή :

- Ο πρώτος τρόπος είναι η αλγεβρική προσέγγιση, με την οποία διαχωρίζεται η βασική πληροφορία από τα πλεονάζοντα bit πληροφορίας με λειτουργίες μεταθέσεων.
- Και ο δεύτερος τρόπος πραγματοποιείται με την χρήση ενός αποκωδικοποιητή ακολουθιακής ακύρωσης SCD (Successive Cancellation Decoder). Αρχικοποιείται προκειμένου να χρησιμοποιηθεί ως κωδικοποιητής έχοντας για είσοδο τον πλεονασμό και τα δεδομένα κωδικοποιημένα. Έτσι αναγκάζεται να υπολογίσει στην έξοδο την βασική πληροφορία αναδρομικά δηλαδή λειτουργεί ανάποδα και αυτό επαληθεύει ότι εξάγει σωστά το ζητούμενο αποτέλεσμα.

Έπειτα από έρευνες και συγκρίσεις, η συστηματική μορφή δείχνει σημαντική βελτίωση σε σχέση με τη μη συστηματική, όπως φαίνεται από προσομοιώσεις χρήσης κωδίκων. Συγκρίθηκαν λοιπόν, οι δύο κώδικες με τα ίδια χαρακτηριστικά δηλαδή ίδιο μέγεθος block για $N = 256$, ρυθμό κώδικα $R = 1/2$ σε ίδιες συνθήκες μετάδοσης όπως, για παράδειγμα, σε ίδιο δίαυλο μετάδοσης, με την ίδια τεχνική αποκωδικοποίησης, τεχνική διαμόρφωσης και επίδρασης θορύβου. Το αποτέλεσμα αυτής της παρατήρησης ήταν σημαντικό κέρδος κατά την διαδικασία της συστηματικής κωδικοποίησης, όπως φαίνεται και στο παρακάτω σχήμα, πετυχαίνοντας σταθερή την τιμή του συνολικού FER (frame error bit), δηλαδή της αναλογίας των πλαισίων προς τα μεταδιδόμενα πλαίσια.



Σχήμα 31: Κέρδος FER σε systematic. Πηγή: <https://www.computerworld.com/>

Η πτώση του BER (bit error rate), από την εικόνα, δείχνει το ποσοστό των n bit που έχουν σφάλματα σε σύγκριση με τον συνολικό αριθμό των bits που λαμβάνονται κατά τη διάρκεια μίας μετάδοσης, ήταν αντίθετη σε σχέση με αυτό που αναμενόταν. Πιθανόν γι' αυτό ευθύνεται η χρήση αποκωδικοποιητή ακολουθιακής ακύρωσης καθώς η συγκεκριμένη τεχνική φαίνεται να ελαχιστοποιεί τα λάθη. Έτσι ο Arıkan, θεωρεί πως ο προτιμότερος τρόπος που θα γίνει η αποκωδικοποίηση στο δέκτη θα πρέπει να είναι με την μέθοδο της ακολουθιακής ακύρωσης. Παράλληλα προτείνονται 3 αλγόριθμοι συστηματικής κωδικοποίησης από τους Vangala, Hong και Viterbo, των οποίων η πολυπλοκότητα συγκρίνεται με την αρχική μη συστηματική άποψη του Arıkan. Χρησιμοποιείται ο αριθμός XOR των πράξεων που χρειάζεται για την υλοποίηση του κυκλώματος.

Ο πρώτος κωδικοποιητής που προτείνεται λειτουργεί ακριβώς όπως ένας κωδικοποιητής μη συστηματικός. Δεν χρησιμοποιεί αναδρομή αλλά διαφορετική σειρά υπολογισμών με ταυτόχρονη χρήση ενδιάμεσης μνήμης. Αποφεύγεται η αναδρομή με χρήση του ίδιου αριθμού XOR πράξεων αλλά με κόστος χρήσης σχεδόν πέντε φορές μεγαλύτερης μνήμης.

Ο δεύτερος κωδικοποιητής χρησιμοποιεί αναδρομή. Η αναδρομή λειτουργεί με βάση την συνεργασία πράξεων XOR και πόρων μνήμης. Στην πράξη της κωδικοποίησης δημιουργείται ένα διάνυσμα και δείκτες διανύσματος, οι οποίοι απλοποιούν την διαδικασία διάσχισης των υπο-δέντρων. Χρησιμοποιείται ανάστροφη ταξινόμηση για την υλοποίηση των δεικτών προκειμένου να ακολουθούν την σειρά του κωδικοποιητή. Τελικά η ολική πραγματοποίηση αυτής της μεθόδου απαιτεί μεγάλο αριθμό πράξεων XOR με λιγότερους πόρους μνήμης.

Η τρίτη πρόταση κωδικοποίησης διαθέτει μια ικανή προσέγγιση σε θέματα ανάγκης πόρων μνήμης. Λειτουργεί με αναδρομή όπως και ο προηγούμενος, με τη διαφορά ότι χρειάζεται μόνο ένα διάνυσμα μήκους N με το μόνο ελάττωμα την αύξηση των απαιτούμενων πράξεων XOR, καθώς απαιτούνται σχεδόν τέσσερις φορές περισσότερες πράξεις από έναν μη συστηματικό κωδικοποιητή.

3.4.3 Αποκωδικοποίηση κατά ARIKAN

Σύμφωνα με τον Arikan, η διαδικασία αποκωδικοποίησης που προτείνεται είναι αυτή της ακολουθιακής ακύρωσης γνωστή και ως SCD (successive cancellation decoding). Είναι τεχνική που χρησιμοποιείται από έναν δέκτη επιτρέποντας την αποκωδικοποίηση δύο ή περισσότερων πακέτων τα οποία έφτασαν ταυτόχρονα. Στον δέκτη λοιπόν ξεκινά η διαδικασία, υποθέτοντας ότι γνωρίζει τα κανάλια που έχουν παγώσει. Ο αποκωδικοποιητής έχει την δυνατότητα να εξάγει πληροφορίες για τα κανάλια που μεταφέρουν την πληροφορία εφόσον γνωρίζει την συμπεριφορά του θορύβου στα παγωμένα κανάλια. Η αποκωδικοποίηση ξεκινά από το κανάλι που διαθέτει το πιο μικρό channel index και συνεχίζει. Χρησιμοποιείται για το κάθε κανάλι το σύνολο εξόδου, η είσοδος του καναλιού και οι προηγούμενοι είσοδοι εφόσον υπάρχουν. Οι είσοδοι των επόμενων καναλιών με μεγαλύτερο channel index θεωρούνται ως θόρυβος. Ο αποκωδικοποιητής απαρτίζεται από N μονάδες απόφασης. Κάθε μία από αυτές τις μονάδες αντιστοιχούν σε ένα από τα κανάλια. Οι μονάδες απόφασης έχουν το εξής χαρακτηριστικό:

- Αν το στοιχείο δεν ανήκει στην πληροφορία είναι δηλαδή παγωμένο bit (frozen), τότε η έξοδος είναι 0 και ο υπολογισμός διαδίδεται στις άλλες μονάδες απόφασης.
- Αν το στοιχείο ανήκει στην πληροφορία, δεν είναι δηλαδή παγωμένο bit, η μονάδα απόφασης ελέγχει αν έχει λάβει υπολογισμούς για όλες τις προηγούμενες εισόδους. Σε περίπτωση που δεν έχει γίνει ο υπολογισμός, περιμένει να ολοκληρώσουν την διαδικασία οι προηγούμενες μονάδες απόφασης.

Ο υπολογισμός μεταδίδεται σε όλες τις υπόλοιπες μονάδες απόφασης, προκειμένου να ληφθούν υπόψιν τα δεδομένα. Είναι αρκετά σημαντικό να έχουν γίνει σωστά οι εκτιμήσεις διότι αν υπάρχει σφάλμα μεταδίδεται και στους επόμενους υπολογισμούς και συνεπώς εξάγονται λανθασμένα αποτελέσματα.

3.4.4 Πλεονέκτηματα – Μειονεκτήματα

Βασικό πλεονέκτημα της τεχνικής Polar coding είναι ότι έχουν μέτρια πολυπλοκότητα κωδικοποίησης και αποκωδικοποίησης, δηλαδή $O(n \log(n))$ και συνεπώς είναι εύκολο να

εφαρμοστούν. Γι' αυτό, είναι κατάλληλοι για πολλές εφαρμογές και μοιάζουν ιδανικοί για την τεχνική του ασύρματου πομπού/δέκτη 5G. Είναι κατάλληλοι για κανάλι ελέγχου όπου το μέγεθος ωφέλιμου φορτίου είναι μικρότερο. Όπως αναφέρθηκε και παραπάνω, ήταν μία από τις μεθόδους συμβολής στην επίτευξη υψηλής απόδοσης στα ασύρματα δίκτυα 5G, που πέτυχε μαζί με την εταιρεία Huawei 27 Gbps. Η πολική κωδικοποίηση έχει την δυνατότητα να εφαρμοστεί στο μπλοκ διαμόρφωσης με τους εξής τρόπους: με άμεση διαμόρφωση, με πολικά πλέγματα και με τεχνικές πολλαπλών επιπέδων.

Η μέθοδος που χρησιμοποιείται στην κωδικοποίηση των Polar κωδικών είναι η SCD(Successive Cancellation Decoder), έχει πολύ αργή απόδοση και συγκεκριμένα σε σύγκριση με τις τεχνικές κωδικοποίησης των Turbo και LDPC θεωρείται η χειρότερη. Προσφέρει υψηλότερη καθυστέρηση, της τάξης $O(N)$. Παρέχουν την καλύτερη απόδοση με προηγμένους αποκωδικοποιητές αλλά το κόστος είναι οι υψηλότερες τιμές σε αντίθεση με άλλους κώδικες.

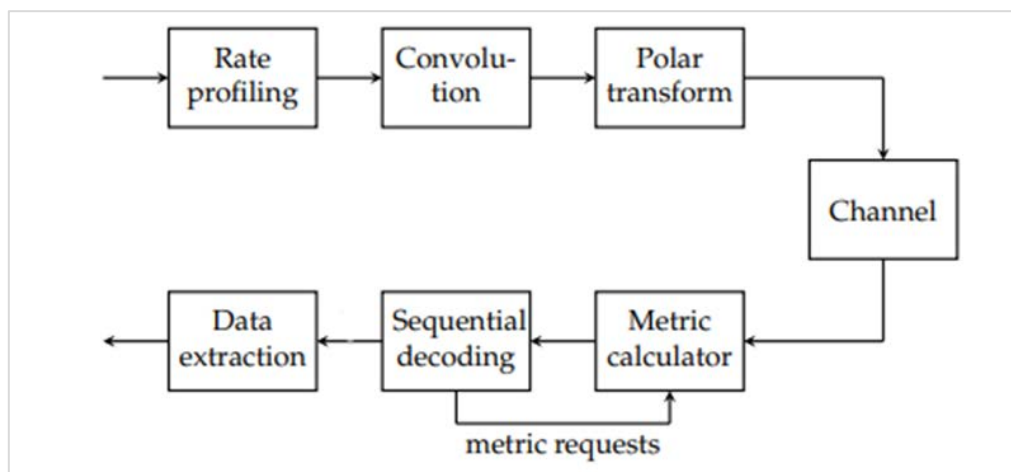
3.4.5 Ανεπτυγμένη Έκδοση SCD

Οι εκδόσεις SCL (successive cancellation list) και SCS (successive cancellation stack) αποκωδικοποίησης είναι βελτιωμένες εκδόσεις του αλγόριθμου SCD και χρησιμοποιούνται για τη βελτίωση της απόδοσης πεπερασμένου μήκους των polar κωδικών. Σε συνδυασμό και των τριών αυτών που προαναφέρθηκαν δημιουργήθηκε ο αλγόριθμος αποκωδικοποίησης Successive Cancellation Hybrid (SCH). Βασικό πλεονέκτημά του είναι ότι έχει την ικανότητα να πετύχει καλύτερη αντιστάθμιση μεταξύ της υπολογιστικής πολυπλοκότητας και της πολυπλοκότητας χώρου. Στόχος είναι επίσης η μείωση της πολυπλοκότητας και αυτό επιτυγχάνεται μέσω μίας τεχνικής κλαδέματος για την αποφυγή εργασιών που είναι περιττές, όπως η αναζήτηση των μονοπατιών. Έπειτα από προσομοιώσεις που πραγματοποιήθηκαν με τις απαραίτητες διαμορφώσεις, η ανάλυση απόδοσης και πολυπλοκότητας, έδειξαν ότι και οι τρεις βελτιωμένοι αλγόριθμοι αποκωδικοποίησης μπορούν να επιτύχουν απόδοση πολύ κοντά στην απόδοση της αποκωδικοποίησης μέγιστης πιθανότητας ML (maximum-likelihood) και με αποδεκτή πολυπλοκότητα. Με την τεχνική λοιπόν του κλαδέματος, η πολυπλοκότητα των αποκωδικοποιητών μπορεί να είναι πολύ κοντά στην πολυπλοκότητα του αποκωδικοποιητή SC σε μέτρια και υψηλή κλίμακα αναλογίας σήματος προς θόρυβο (SNR).

3.4.6 Κώδικας PAC

Ο Arikan εισήγαγε πρόσφατα μία καινούρια μέθοδο πολικής κωδικοποίησης που ονομάζεται PAC-polarization-adjusted convolutional (συνελικτικός κώδικας προσαρμοσμένης πόλωσης).

Τέτοιου είδους κωδικοί υπερτερούν ακόμη και των CRC (Cyclic redundancy check) συμβατικών πολικών κωδίκων. Η βασική ιδέα βασίζεται στην αντικατάσταση των CRC κωδικών και αντικατάσταση της αποκωδικοποίησης λίστας με διαδοχική αποκωδικοποίηση. Στην κωδικοποίηση ο Arıkan χρησιμοποιεί έναν συνελκτικό προ-κωδικοποιητή rate-1, ο οποίος συνδέεται με έναν τυπικό πολικό κωδικοποιητή. Μόνο k από τα n bits της εισόδου στον συνελκτικό προ-κωδικοποιητή μεταφέρουν τα δεδομένα που είναι το διάνυσμα d . Τα υπόλοιπα $n-k$ της εισόδου ορίζονται στο 0. Όπως και στους υπόλοιπους συμβατικούς πολικούς κώδικες, η απόδοση του κώδικα PAC κρίνεται από τις θέσεις της εισόδου που μεταφέρουν τα δεδομένα, οι οποίες έχουν την τιμή 0. Η επιλογή παγωμένων θέσεων στο 0 έχει ονομαστεί από τον Arıkan rate-profiling. Σε σύγκριση με άλλους κώδικες polar, η επιλογή των παγωμένων θέσεων δεν είναι γνωστή. Η βασική λειτουργία του αποκωδικοποιητή είναι η διαδοχική αποκωδικοποίηση. Λαμβάνονται οι μετρήσεις διαδρομής που χρησιμοποιούνται από τον διαδοχικό αποκωδικοποιητή μέσω επαναλαμβανόμενων κλήσεων στον αποκωδικοποιητή διαδοχικής ακύρωσης για τον πολικό κώδικα. Οι κωδικοί PAC είναι κωδικοί δέντρων και μπορούν να αποκωδικοποιηθούν με την χρήση αλγορίθμων αναζήτησης δέντρων. Θεωρώντας τη λίστα αποκωδικοποίησης, η οποία ανταλλάσσει μία σταθερή πολυπλοκότητα χρόνου για μεγάλη απαίτηση μνήμης που την χρησιμοποιεί για να αποθηκεύσει λίστες μονοπατιών. Στη συνέχεια, χρησιμοποιείται η μέθοδος αποκωδικοποίησης Fano με μεταβλητή πολυπλοκότητα χρόνου, όμως είναι πολύ αποδοτική στη μνήμη. Η αποκωδικοποίηση στους συνελκτικούς κώδικες ονομάζεται M-algorithm. Μερικά αποτελέσματα κάνουν χρήση λίστας. Η αποκωδικοποίηση παρουσιάστηκε πρώτη φορά από την Huawei το 2019. Μετά χρησιμοποιήθηκε και στους κώδικες PAC.



Σχήμα 32: Ροή διαδικασίας και εξαγωγή δεδομένων. Πηγή: Ιδία Επεξεργασία

3.4.7 Μελλοντικές Επεκτάσεις

Έπειτα την αρχική ιδέα του Arikan, όλη η ερευνητική κοινότητα προσπάθησε να βελτιώσει τα χαρακτηριστικά του αλγορίθμου. Έτσι πέρα από την αποκωδικοποίηση της ακολουθιακής ακύρωσης εξετάστηκε και η χρήση αποκωδικοποιητή διάδοσης πεποίθησης (Belief Propagation Decoder), τεχνική που χρησιμοποιείται για το δίκτυο 5G. Οι συγκεκριμένοι αποκωδικοποιητές υπολογίζουν την οριακή κατανομή για οποιοδήποτε σύμβολο άγνωστο, σε συνάρτηση με τα γνωστά έως τώρα σύμβολα. Αποκωδικοποιητές ακολουθιακής ακύρωσης έχουν χαμηλότερη πολυπλοκότητα από τους αποκωδικοποιητές διάδοσης πεποίθησης και αυτό τους καθιστά αποκωδικοποιητές με την βέλτιστη απόδοση. Προτείνεται επίσης από τους C. Xing, B. Wang και S. Zhao αποκωδικοποιητής με χαμηλότερη τιμή πολυπλοκότητας ο οποίος είναι ένας συγκερασμός των ιδεών του Arikan για τους Polar κώδικες και του Gallager για τους LDPC κώδικες. Ειδικότερα ο ερευνητής Gallager, πραγματοποίησε τους υπολογισμούς του με την βοήθεια εκθετικών και λογαριθμικών πράξεων. Οι πράξεις αυτές απλοποιούνται με την βοήθεια προσεγγίσεων καταλήγοντας τελικά σε έναν αποκωδικοποιητή χαμηλής πολυπλοκότητας.

3.5 LDPC Κώδικας

Σύμφωνα με τις ιστοσελίδες <https://www.dsprelated.com/>, <https://eprints.soton.ac.uk/>, <https://www.researchgate.net/> και <https://docplayer.gr/> δύνανται να ειπωθούν οι παρακάτω αναφορές σχετικά με τον κώδικα LDPC.

Ο κώδικας ελέγχου ισοτιμίας χαμηλής πυκνότητας (LDPC) είναι ένας γραμμικός κώδικας διόρθωσης σφαλμάτων ιδανικός για μεγάλα μεγέθη block κώδικα και μετάδοσης μηνύματος μέσω ενός θορυβώδους καναλιού. Το όνομα τους σχετίζεται με το χαρακτηριστικό της μήτρας ελέγχου ισοτιμίας H και διαθέτει μόνο ορισμένους άσσους συγκριτικά με την ποσότητα των μηδενικών του πίνακα. Βασικό πλεονέκτημά τους, είναι ότι έχουν απόδοση η οποία είναι κοντά στην χωρητικότητα διαφορετικών καναλιών. Οι κώδικες LDPC είναι γνωστοί και ως κώδικες Gallager, καθώς αναπτύχθηκαν από τον Robert G. Gallager κατά την διάρκεια της διδακτορικής του διατριβής στο Τεχνολογικό Ινστιτούτο της Μασαχουσέτης το 1960. Οι κώδικες χρησιμοποιήθηκαν στα τέλη του 1990 για δορυφορικές επικοινωνίες και εφαρμογές στο Deep Space Network. Σημειώθηκε πρόοδος στους κώδικες ελέγχου ισοτιμίας χαμηλής πυκνότητας σε σημείο που ξεπέρασαν και τους κώδικες Turbo σχετικά με το επίπεδο σφάλματος και την απόδοση στο εύρος υψηλότερου ρυθμού κώδικα, με αποτέλεσμα οι Turbo κώδικες να χρησιμοποιούνται για χαμηλότερους ρυθμούς.

3.5.1 Τρόποι Αναπαράστασης LDPC Κώδικα

3.5.1.1 Πίνακας H

Οι LDPC κώδικες όπως αναφέραμε και παραπάνω είναι γραμμικοί block κώδικες και μπορούν να αναπαρασταθούν με την χρήση ενός πίνακα ελέγχου ισοτιμίας. Ένας κώδικας LDPC θεωρείται ως διπλός (dual) ή μηδενικός (null) πίνακας ισοτιμίας $m \times n$. Σημαντικά στοιχεία για την αναπαράσταση ενός LDPC κώδικα μέσω του πίνακα H είναι τα εξής:

1. Η παράμετρος m είναι το πλήθος των εξισώσεων ελέγχου ισοτιμίας. Όσο αυξάνεται η τιμή της τόσο μεγαλώνει η πιθανότητα για διόρθωση σφαλμάτων του κώδικα.
2. Η παράμετρος n ορίζεται ως το μέγεθος block του κώδικα και η αύξηση της τιμής της βελτιώνει τον κώδικα.
3. Η παράμετρος Rank καθορίζει τον ρυθμό του πίνακα. Είναι ο αριθμός των ανεξάρτητων γραμμών που διαθέτει ο πίνακας.
4. Τα βάρη των γραμμών και των στηλών του πίνακα. Ορίζεται ως το πλήθος των στοιχείων του πίνακα που είναι μη μηδενικά.

Για παράδειγμα στην παρακάτω εικόνα παρατηρούμε ότι ο πίνακας διαθέτει $n=10$ στήλες, εκ των οποίων οι 7 έχουν βάρος 2 (δηλαδή τα μη μηδενικά στοιχεία είναι 2) και οι υπόλοιπες 3 στήλες έχουν βάρος 3. Επίσης, ο πίνακας έχει $m=5$ γραμμές, με τις 2 να έχουν βάρος 4 και οι 3 με βάρος 5.

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

Οι LDPC κώδικες που έχουν σταθερό βάρος γραμμών και στηλών ονομάζονται κανονικοί κώδικες (regular). Οι LDPC κώδικες που δεν έχουν σταθερά βάρη γραμμών και στηλών ονομάζονται μη κανονικοί.

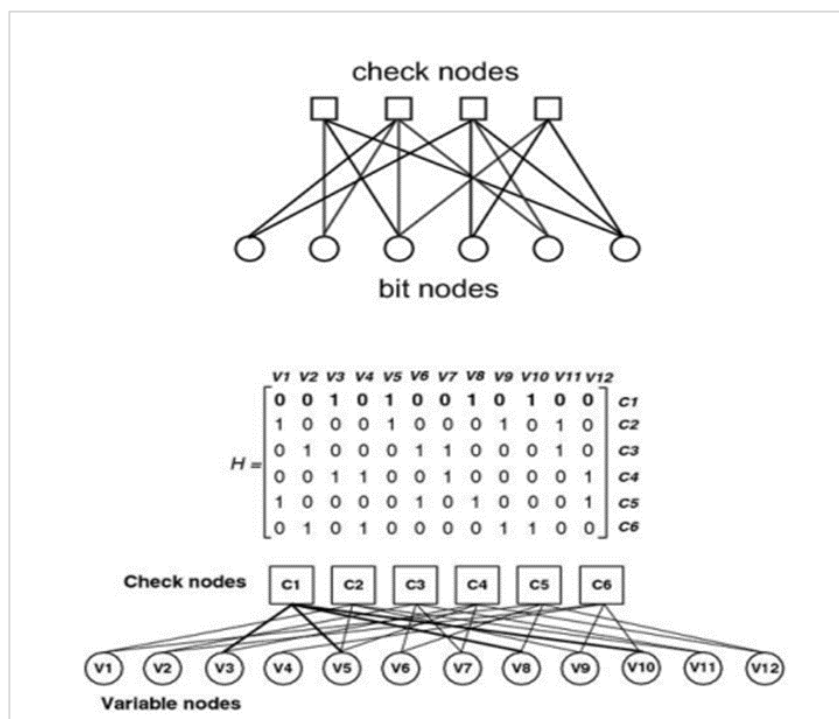
3.5.1.2 Tanner Graph

Οι κώδικες LDPC απεικονίστηκαν γραφικά από τον Tanner το 1981. Ο ίδιος προσπάθησε να αναπτύξει μία αναδρομική προσέγγιση έτσι ώστε να επιτευχθεί η κατασκευή μεγάλου μήκους ισχυρών κωδίκων με την βοήθεια μικρότερων και πιο απλών. Υποστήριξε λοιπόν ότι οι διαδικασίες κωδικοποίησης και αποκωδικοποίησης περιγράφονται από την θεωρία των γράφων. Το σημαντικό πλεονέκτημα αυτής την θεωρίας για την χρήση αναδρομικότητας είναι

η μείωση την πολυπλοκότητας, ένα από τα πιο σημαντικά χαρακτηριστικά για την υλοποίηση του κώδικα.

Όμως, το τροχοπέδη σε αυτή τη μέθοδο είναι ότι η αναδρομικότητα δεν επιτρέπει στους κώδικες να αποδίδουν τα μέγιστα, με αποτέλεσμα να υπολειτουργεί, διότι ο κώδικας αντιμετωπίζεται ως κομμάτια μικρών κωδίκων μικρής ισχύος και όχι ως ένα ενιαίο κομμάτι κώδικα. Ο Tanner στην ανάλυσή του υποστήριξε την θεωρία των διμερών γραφημάτων. Είναι τα γραφήματα με δύο ομάδες κόμβων οι οποίες συμβολίζονται με C (check nodes) και V (bit nodes). Κάθε κόμβος από το σύνολο C μπορεί να συνδεθεί με κόμβους μόνο από το σύνολο V.

Η σύνδεση των κόμβων πραγματοποιείται με την χρήση συνδέσεων. Με αυτόν τον τρόπο μπορεί να μετατραπεί ένα μεγάλου μήκους κώδικα σε δύο ή περισσότερους κώδικες με μικρότερο μήκος. Παρακάτω στην φωτογραφία απεικονίζεται ένα γράφημα Tanner στο οποίο γίνεται διαχωρισμός σε subcode κόμβους και σε digit κόμβους. Οι subcode κόμβοι αναπαριστούν τις σειρές του κώδικα ενώ οι digit κόμβοι τις στήλες (μήτρα ελέγχου ισοτιμίας). Ένας κόμβος subcode συνδέεται με έναν κόμβο digit μόνο αν υπάρχει μια μη μηδενική τιμή στην αντίστοιχη διασταύρωση γραμμής και στήλης.



Σχήμα 33: Subcode κόμβοι. Πηγή: <https://www.dsprelated.com/>

3.5.1.3 Κατασκευή Πίνακα PARITY CHECK H

Η παραπάνω διαδικασία αναπαράστασης, που αναφέρθηκε στο κεφάλαιο 6.2.1, από τον Gallager προορίζεται για κανονικούς κώδικες LDPC (regular), δηλαδή για κώδικες με σταθερό αριθμό μη μηδενικών στοιχείων στις γραμμές και στις στήλες. Ο πίνακας ισοτιμίας H δημιουργήθηκε ως αλληλουχία υπο-πινάκων έχοντας σε κάθε σειρά και στήλη σταθερά στοιχεία. Οι μη κανονικοί LDPC κώδικες ορίζονται όπως οι κανονικοί αλλά ο αριθμός των μη μηδενικών στοιχείων είναι διαφορετικός σε κάθε γραμμή και στήλη.

$$H = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 1 & 0 \\ 1 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 \end{bmatrix}.$$

Σύμφωνα με την μέθοδο κατασκευής των Neal και MacKay ο πίνακας κατασκευάζεται σταδιακά. Η διαδικασία περιλαμβάνει πρόσθεση μίας στήλης από αριστερά. Οι θέσεις που θα λάβουν τα μη μηδενικά στοιχεία επιλέγονται τυχαία με βάση τις προδιαγραφές της κατανομής. Οι στήλες συνεχίζουν να προστίθενται μέχρι να επιτευχθεί ο επιθυμητός αριθμός μη μηδενικών στοιχείων σε κάθε σειρά του πίνακα.

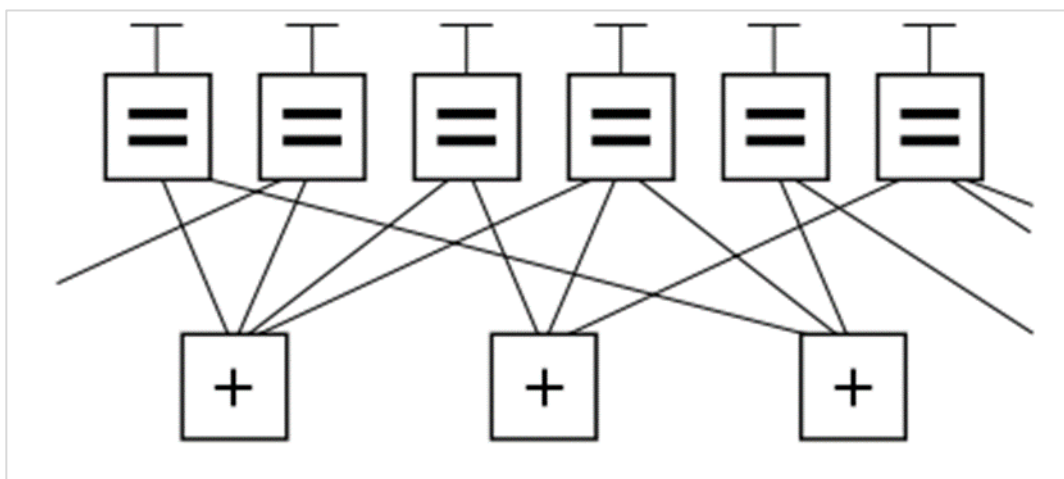
3.5.2 Κωδικοποίηση LDPC

Η κωδικοποίηση σχετίζεται με την παραγωγή κωδικών λέξεων του κώδικα, από τα μηνύματα πληροφορίας. Θα μελετηθεί ο τρόπος με τον οποίο παράγεται μια σειρά στοιχείων (n) από μία άλλη σειρά στοιχείων (k), για την δημιουργία ενός LDPC (n, k) κώδικα με ρυθμό $R = k / n$. Το σύμβολο $\oplus$ είναι η πρόσθεση modulo-2, δηλαδή η πράξη XOR. Οι LDPC κώδικες είναι γραμμικοί block κώδικες. Στην προσπάθεια κωδικοποίησης χρησιμοποιείται ένας συνηθισμένος γραμμικός κώδικας, με την βοήθεια ενός πίνακα γεννήτριας G ή του πίνακα ισοτιμίας H. Η πολυπλοκότητα θα είναι τετραγωνικού χρόνου $O(n^2)$, αυτό σημαίνει ότι όσο αυξάνεται το μέγεθος n του block αυτό θα αυξάνεται τετραγωνικά. Έχει αναφερθεί ξανά, ότι η πολυπλοκότητα του κώδικα αποτελεί ένα πολύ σημαντικό στοιχείο καθώς επηρεάζει και την ταχύτητα της διαδικασίας της κωδικοποίησης. Με την βοήθεια του πίνακα H και του χαρακτηριστικού της αραιότητας υπάρχει δυνατότητα να επιτευχθεί μικρότερη

πολυπλοκότητα της τάξης $O(n)$. Γενικότερα, ανεξάρτητα από τον τρόπο κωδικοποίησης, η διαδικασία αναλύεται σε δύο στάδια που είναι τα εξής:

1. Το στάδιο επεξεργασίας, περιλαμβάνει offline υπολογισμούς προκειμένου να λάβουν οι πίνακες την μορφή που επιθυμεί ο αναλυτής. Δεν επηρεάζει καθόλου η διαδικασία αυτή τα στοιχεία που θα κωδικοποιηθούν και πραγματοποιείται μόνο μία φορά.
2. Το στάδιο που γίνεται η κωδικοποίηση και αποτελείται από online υπολογισμούς. Εδώ παράγονται οι κωδικές λέξεις και σε αντίθεση με το προηγούμενο στάδιο υπάρχει επαφή με τα δεδομένα που κωδικοποιούνται.

Οι LDPC κώδικες ορίζονται από έναν αραιό πίνακα, τον πίνακα ελέγχου ισοτιμίας. Ο συγκεκριμένος πίνακας συνήθως είναι τυχαίος, με εξαίρεση μερικές φορές τους περιορισμούς αραιότητας. Παρακάτω υπάρχει μία φωτογραφία που απεικονίζει τμήμα γραφήματος κώδικα LDPC. Υπάρχουν λοιπόν, n μεταβλητοί κόμβοι στην πάνω πλευρά του γραφήματος που συνδέονται με $n-k$ κόμβους περιορισμούς, στην κάτω πλευρά της εικόνας. Η συγκεκριμένη αναπαράσταση ενός (n, k) κώδικα LDPC είναι η πιο δημοφιλής. Τα στοιχεία bits που αποτελούν το κομμάτι του έγκυρου μηνύματος, έχουν την θέση στο T, στην κορυφή των κόμβων της πάνω πλευράς του γραφήματος και ικανοποιούν τους γραφικούς περιορισμούς. Αγνοείτε τις γραμμές των δύο ακριανών κόμβων που βγαίνουν έξω από το γράφημα.



Σχήμα 34: τμήμα γραφήματος κώδικα LDPC. Πηγή: Ιδία Επεξεργασία

Ειδικότερα, οι γραμμές που συνδέονται με έναν κόμβο που φέρει το σύμβολο "=" έχουν την ίδια τιμή και αντίστοιχα οι γραμμές που συνδέονται με έναν κόμβο με σύμβολο "+" πρέπει να αθροίζονται με τη πράξη modulo-2 στο μηδέν. Αυτός είναι ένας $(6,3)$ γραμμικός κώδικας με $n=6$ και $k=3$. Υπάρχουν 8 πιθανές τιμές 6 bit που είναι οι έγκυρες κωδικές λέξεις. Ουσιαστικά το κομμάτι του LDPC κώδικα αναπαριστά ένα μήνυμα που αποτελείται από 3 bit,

κωδικοποιημένο σαν να είναι 6 bit. Είναι οι εξής τιμές : 000000, 011001, 110010, 101011, 111100, 100101, 001110, 01011. Η σημασία του πλεονασμού που χρησιμοποιείται είναι για να αυξήσει την πιθανότητα να ανακτήσει την αρχική (σωστή πληροφορία) από τυχόν παρεμβολές και σφάλματα του καναλιού.

$$\mathbf{H} = \begin{pmatrix} 1 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 1 \\ 1 & 0 & 0 & 1 & 1 & 0 \end{pmatrix}.$$

Στον πίνακα H, παραπάνω οι σειρές αντιπροσωπεύουν έναν από τους τρεις περιορισμούς ελέγχου ισοτιμίας και οι στήλες ένα από τα 6 bit της κωδικής λέξης που λαμβάνεται.

$$\mathbf{H} = \begin{pmatrix} 1 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 1 \\ 1 & 0 & 0 & 1 & 1 & 0 \end{pmatrix}_1 \sim \begin{pmatrix} 1 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 1 \\ 0 & 1 & 1 & 0 & 1 & 0 \end{pmatrix}_2$$

$$\sim \begin{pmatrix} 1 & 1 & 1 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 1 & 0 & 1 \end{pmatrix}_3 \sim \begin{pmatrix} 1 & 1 & 1 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 0 & 0 & 1 \end{pmatrix}_4.$$

Τα βήματα που ακολουθούν παραπάνω είναι τα εξής:

1. Αρχικά υπάρχει ο πίνακας H.
2. Προστίθεται η πρώτη σειρά στη τρίτη.
3. Οι σειρές 2 και 3 εναλλάσσονται.
4. Προστίθεται η πρώτη σειρά στη τρίτη σειρά.

Από αυτή τη διαδικασία προκύπτει τελικά ο πίνακας γεννήτριας G:

$$\mathbf{G} = \begin{pmatrix} 1 & 0 & 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 1 & 1 & 1 \\ 0 & 0 & 1 & 1 & 1 & 0 \end{pmatrix}.$$

Έπειτα πολλαπλασιάζονται και οι 8 πιθανές συμβολοσειρές από 3 bit με τον πίνακα G και προκύπτουν 8 έγκυρες κωδικές λέξεις. Για παράδειγμα, η κωδική λέξη για την συμβολοσειρά bit 101 είναι:

$$(1 \ 0 \ 1) \odot \begin{pmatrix} 1 & 0 & 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 1 & 1 & 1 \\ 0 & 0 & 1 & 1 & 1 & 0 \end{pmatrix} = (1 \ 0 \ 1 \ 0 \ 1 \ 1),$$

Όταν πραγματοποιείται η κωδικοποίηση ενός πλαισίου, τα στοιχεία (bit) εισόδου επαναλαμβάνονται και μοιράζονται σε ένα σύνολο κωδικοποιητών, γνωστοί και ως συσσωρευτές. Κάθε συσσωρευτής χρησιμοποιείται για τη δημιουργία ενός συμβόλου ισοτιμίας. Ένα αντίγραφο των αρχικών δεδομένων μεταδίδεται με τα bit ισοτιμίας προκειμένου να δημιουργηθούν τα σύμβολα του κώδικα.

Σε αντίθεση οι κλασσικοί κώδικες Turbo χρησιμοποιούν δύο κωδικούς διαμορφωμένους παράλληλα, καθένας από αυτούς κωδικοποιεί ολόκληρο το μπλοκ εισόδου δεδομένων. Οι κωδικοποιητές αυτοί είναι αναδρομικοί συνελκτικοί κώδικες που διαχωρίζονται από έναν κώδικα που διαθέτει αντίγραφο του πλαισίου.

Ο LDPC κώδικας χρησιμοποιεί αρκετούς συσσωρευτές παράλληλα. Καθένας από τους συσσωρευτές κωδικοποιεί μόνο ένα μικρό τμήμα του πλαισίου εισόδου. Εξαιτίας του επιπέδου πλεονασμού για κάθε bit εισόδου και της δυνατότητας ακριβέστερης διαχείρισης των συνδέσεων διαφόρων κωδικών στοιχείων οι κώδικες LDPC προσφέρουν μεγαλύτερη ευελιξία από του κώδικες Turbo. Βέβαια οι κώδικες Turbo δείχνουν να αποδίδουν καλύτερα σε χαμηλούς ρυθμούς κώδικα. Σχετικά με τους συσσωρευτές, το πρώτο σύνολο bit ισοτιμίας που χρησιμοποιείται και αποθηκεύεται κατά την διαδικασία της κωδικοποίησης, χρησιμοποιείται και για την δημιουργία του επόμενου συνόλου bit ισοτιμίας.

3.5.3 Αποκωδικοποίηση LDPC

Κατά την διαδικασία της αποκωδικοποίησης (decoding), της παραγωγής δηλαδή μηνυμάτων πληροφορίας από τις κωδικές λέξεις που έχουν ληφθεί, χρησιμοποιούνται οι αλγόριθμοι που ονομάζονται message passing algorithms. Λόγω παρουσίας θορύβου στο κανάλι υπάρχει πιθανότητα ένα ή παραπάνω bits να έχουν αλλάξει τιμή σε αντίθεση με τα αρχικά bit που έχουν αποσταλεί. Οι αλγόριθμοι λοιπόν, βασίζονται στην μετάδοσης της πληροφορίας στους κόμβους με την βοήθεια της θεωρίας των γράφων. Οι κόμβοι του γράφου εργάζονται στα δεδομένα που περιέχονται στους κόμβους. Οι message passing αλγόριθμοι ονομάζονται και επαναληπτικοί κώδικες καθώς η πληροφορία κινείται ανάμεσα στους κόμβους επαναληπτικά μέχρι να προκύψει το ανάλογο αποτέλεσμα σχετικά με την συνθήκη που έχει οριστεί ως τελική. Το χαρακτηριστικό της επανάληψης στην αποκωδικοποίηση είναι το σημαντικότερο στοιχείο των LDPC κωδικών διότι μειώνει αρκετά την τιμή της πολυπλοκότητας, η οποία είναι

γραμμικού χρόνου. Αυτό την καθιστά ιδανική σε υλοποίηση συστημάτων με μεγάλα block length. Σε μερικούς από αυτούς τους αλγόριθμους, των οποίων οι τιμές των bits μπορούν να έχουν τιμή 0 ή 1, η πληροφορία που κυκλοφορεί ανάμεσα στους κόμβους είναι τιμές πιθανοτήτων για τον αν η τιμή είναι αληθής ή όχι. Οι τιμές αυτές, είναι αλγόριθμοι που ονομάζονται belief propagation decoding algorithms και οι πιθανότητες εκφράζονται ως λόγοι λογαρίθμων. Όταν η χρήση λογαρίθμων επιτρέπει την μετατροπή της πράξης του πολλαπλασιασμού σε πρόσθεση οι πιθανότητες εκφράζονται ως λόγοι λογαρίθμων και οι αλγόριθμοι ονομάζονται sum product decoding algorithms.

3.5.3.1 Επαναληπτική, MESSAGE-PASSING Αποκωδικοποίηση

Η συγκεκριμένη διαδικασία αποκωδικοποίησης δεν αφορά μόνο τους LDPC κώδικες αλλά όλους τους κώδικες που προκύπτουν ως αποτέλεσμα σύνδεσης επιμέρους υπο-κωδικών. Πρόκειται για ένα σύνολο από αποκωδικοποιητές, με χαμηλή πολυπλοκότητα που λειτουργεί με έναν διανεμημένο τρόπο. Στον αλγόριθμο message passing, οι πληροφορίες ανταλλάσσονται μεταξύ των κόμβων. Στα κανάλια τα bit πληροφορίας μπορεί να μεταδοθούν στον δέκτη σωστά ή λανθασμένα. Ο σκοπός του αποκωδικοποιητή είναι να εντοπίσει τα κατεστραμμένα bits και να τα διορθώσει. Διαθέτει μία εξίσωση που ικανοποιεί τον έλεγχο ισοτιμίας για κάθε κωδική λέξη. Αν εισαχθεί λάθος bit η εξίσωση παύει να ικανοποιείται και τότε είναι η σειρά του αποκωδικοποιητή με την επαναληπτική μέθοδο να ανιχνεύσει και να διορθώσει τα λάθη. Μέρος σε αυτήν την διαδικασία, παίρνουν μόνο οι κόμβοι και οι εξισώσεις ελέγχου ισοτιμίας των κόμβων δεν ικανοποιούνται. Στην πράξη, κάθε κόμβος i προωθεί το ίδιο εξερχόμενο μήνυμα M_i , προς όλους τους check κόμβους, με τους οποίους συνδέεται. Η τιμή του μπορεί να είναι 0 ή 1 ή x αν η τιμή είναι διαφορετική του 0 ή του 1, δηλαδή να έχει καταστραφεί κατά την διαδικασία της μετάδοσης. Σε περίπτωση που κάποιος κόμβος λάβει την τιμή x , με την βοήθεια της εξίσωσης ελέγχου ισοτιμίας μπορεί να διορθώσει την τιμή. Οι κόμβοι που δέχονται την πληροφορία στέλνουν πίσω την τιμή στην οποία κατέληξαν στους κόμβους με τους οποίους συνδέονται και που έστειλαν αρχικά την πληροφορία. Σε περίπτωση που ο κόμβος έχει κατεστραμμένη τιμή και λάβει εκ νέου τιμή 1 ή 0, ανανεώνει την τιμή του με την καινούργια. Αυτό επαναλαμβάνεται μέχρι να μην υπάρχουν κατεστραμμένα bits ή μέχρι έναν ορισμένο μέγιστο αριθμό επαναλήψεων.

$$\begin{aligned}
 H &= \begin{bmatrix} 1 & 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & 1 & 0 & 1 \end{bmatrix} \\
 B &= \begin{bmatrix} 1 & 2 & 4 \\ 2 & 3 & 5 \\ 1 & 5 & 6 \\ 3 & 4 & 6 \end{bmatrix} \\
 A &= \begin{bmatrix} 1 & 3 \\ 1 & 3 \\ 2 & 4 \\ 1 & 4 \\ 2 & 3 \\ 3 & 4 \end{bmatrix} \\
 c &= [0 \ 0 \ 1 \ 0 \ 1 \ 1] \\
 r &= [0 \ 0 \ 1 \ x \ x \ x]
 \end{aligned}$$

Έστω ο παραπάνω πίνακας H για τυχαίο κώδικα και B ο πίνακας που συμβολίζει τις συνδέσεις ανάμεσα σε κόμβους που στέλνουν bit πληροφορίας και κόμβους που δέχονται. Ο πίνακας A αντίστοιχα αναπαριστά την πληροφορία που στέλνεται από τους κόμβους που δέχτηκαν την πληροφορία στους κόμβους που έστειλαν την πληροφορία. Η c είναι η κωδικοποιημένη λέξη και η r αυτή που έφτασε. Αρχικά, θα γίνει η αρχικοποίηση του μηνύματος $M = [0 \ 0 \ 1 \ x \ x \ x]$ που φτάνει. Έπειτα, ο υπολογισμός της ισοτιμίας σε κάθε κόμβο που δέχεται πληροφορία. Ο πρώτος κόμβος συνδέεται με τους 1, 2 και 4 bit κόμβους και έχει ένα bit κατεστραμμένο από τον 4ο. Άρα, υπολογίζοντας $M_1 \oplus M_2 = 0 \oplus 0 = 0$. Θέτει λοιπόν, την τιμή του 4ου bit κόμβου ίση με το μηδέν. Ο δεύτερος κόμβος έχει, επίσης, στα εισερχόμενα μηνύματα του ένα κατεστραμμένο bit το 5ο. Υπολογίζοντας λοιπόν $E_{2,5} = M_2 \oplus M_3 = 0 \oplus 1 = 1$. Έτσι λοιπόν, θέτει την τιμή του 5ου bit ίση με το 1. Η διαδικασία συνεχίζεται και για τους υπόλοιπους κόμβους. Στον 3ο κόμβο υπάρχουν δύο μηνύματα εισερχόμενα τα οποία είναι κατεστραμμένα, είναι το 5ο και το 6ο. Κάθε κόμβος που στέλνει την πληροφορία και έχει κατεστραμμένο μήνυμα υπολογίζει την απάντηση των κόμβων που την λαμβάνουν για να ανανεώσει την τιμή του. Το 4ο bit έχει κατεστραμμένη τιμή και δέχεται 0 και x . Οπότε θα πάρει την τιμή 0. Το ίδιο θα συμβεί και με τον 5ο κόμβο όπου δέχεται τις τιμές 1 και x και τελικά ανανεώνει την τιμή του με 1. Ο 6ος κόμβος δέχεται όλες τις αποκρίσεις με μη ορισμένα μηνύματα και συνεπώς δεν μπορεί να λάβει απόφαση. Τελικά η λέξη θα είναι :

$$r=[0\ 0\ 1\ 0\ 1\ x]$$

με έναν κόμβο κατεστραμμένο. Η διαδικασία λοιπόν θα ξεκινήσει από την αρχή με τις καινούργιες τιμές της λέξης. Μετά την νέα επανάληψη ο αλγόριθμος της αποκωδικοποίησης καταφέρνει να ανακτήσει την αρχική λέξη που είχε σταλεί. Η λέξη είναι η εξής:

$$r=[0\ 0\ 1\ 0\ 1\ 1]$$

3.5.3.2 BIT-FLIPPING Αλγόριθμος

Ο αλγόριθμος bit-flipping βασίζεται στην επαναληπτική message-passing αποκωδικοποίηση. Είναι ένας hard decision message passing αλγόριθμος που χρησιμοποιείται στην αποκωδικοποίηση των LDPC κωδικών. Για κάθε bit πληροφορίας που διακινείται μέσω του γραφήματος Tanner, είναι τιμές 0 ή 1 (δυαδικά μηνύματα), ο ανιχνευτής detector καλείται να αποφασίσει την τιμή παρατηρώντας το λαμβανόμενο διάνυσμα r , του οποία τα στοιχεία είναι πραγματικοί αριθμοί. Έπειτα, η τιμή που θα επιλεγεί περνά στον αποκωδικοποιητή και σε αυτόν τον αλγόριθμο τα μηνύματα διακινούνται ανάμεσα στους κόμβους με την συμβολή αποκλειστικά των έγκυρων συνδέσεων. Οι bit κόμβοι στέλνουν ένα μήνυμα στους check κόμβους με τους οποίους συνδέονται δηλώνοντας αν η τιμή είναι 0 ή 1 και κάθε check κόμβος απαντάει με την βοήθεια της πληροφορίας που έχει συγκεντρώσει. Αποφασίζει για την τιμή χρησιμοποιώντας την εξίσωση ισοτιμίας, δηλαδή το άθροισμα των τιμών των bit που λαμβάνει πρέπει να είναι ίσο με μηδέν σε modulo-2.

Σε περίπτωση που τα περισσότερα μηνύματα που λαμβάνει ένας bit κόμβος έχουν διαφορετική τιμή από την αρχική, τότε η τιμή που έχει ανανεώνεται με την τιμή των check κόμβων. Η συγκεκριμένη διαδικασία επαναλαμβάνεται προκειμένου να ικανοποιηθούν οι εξισώσεις ισοτιμίας σε όλους τους κόμβους ή έστω μέχρι να πραγματοποιηθεί ένα μεγάλο μέρος των επαναλήψεων. Στον αλγόριθμο αυτόν, οι επαναλήψεις σταματούν κατευθείαν μόλις βρεθεί έγκυρη λύση πληρώντας τις προϋποθέσεις της εξίσωσης ελέγχου ισοτιμίας. Τα πλεονεκτήματα αυτού του αλγορίθμου είναι ότι η διαδικασία της επανάληψης σταματάει όταν βρεθεί η λύση και η περίπτωση απόκλισης σε μία έγκυρη λέξη πάντα εντοπίζεται. Ο αλγόριθμος βασίζεται στην αρχή ότι όταν ένα bit εμπλέκεται σε πολλούς check κόμβους, οι οποίοι δεν πληρούν τα κριτήρια της ισοτιμίας, οι πιθανότητες να είναι αυτό το συγκεκριμένο bit λανθασμένο αυξάνονται. Οι LDPC πίνακες κωδικοποίησης, είναι αραιοί δηλαδή τα message bits μοιράζονται στους check κόμβους με τέτοια διασπορά ώστε να είναι εύκολο να εντοπισθούν τα λανθασμένα bits και να διορθωθούν.

Παρακάτω παρουσιάζεται η διαδικασία αποκωδικοποίησης :

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

Βήμα πρώτο είναι η αρχικοποίηση των τιμών: $M = [1 \ 0 \ 1 \ 0 \ 1 \ 1]$.

$$B = \begin{bmatrix} 1 & 2 & 4 \\ 2 & 3 & 5 \\ 1 & 5 & 6 \\ 3 & 4 & 6 \end{bmatrix} \quad A = \begin{bmatrix} 1 & 3 \\ 1 & 3 \\ 2 & 4 \\ 1 & 4 \\ 2 & 3 \\ 3 & 4 \end{bmatrix}$$

Έπειτα, υπολογίζονται τα μηνύματα των check κόμβων. Ο πρώτος κόμβος συνδέεται με τους bit κόμβους 1, 2 και 4. Σύμφωνα με την εξίσωση ελέγχου ισοτιμίας τα μηνύματα του πρώτου check κόμβου είναι $E_{1,1} = M_2 \oplus M_4 = 0 \oplus 0 = 0$. Ομοίως: $E_{1,2} = M_1 \oplus M_4 = 1 \oplus 0 = 1$ και $E_{1,4} = M_1 \oplus M_2 = 1 \oplus 0 = 1$. Ακολουθεί η ίδια διαδικασία και για τον δεύτερο check κόμβο : Συνδέεται με τους bit κόμβους 2, 3, 5. Άρα, $E_{2,2} = M_5 \oplus M_3 = 1 \oplus 1 = 0$, $E_{2,3} = M_2 \oplus M_5 = 0 \oplus 1 = 1$, $E_{2,5} = M_2 \oplus M_3 = 0 \oplus 1 = 1$. Εκτελώντας την ίδια διαδικασία και για τους υπόλοιπους κόμβους προκύπτουν τα αποτελέσματα : $E_{3,1} = 0$, $E_{3,5} = 0$, $E_{3,6} = 0$, $E_{4,3} = 1$, $E_{4,4} = 0$, $E_{4,6} = 1$. Στη συνέχεια, η πληροφορία από τους check κόμβους συλλέγεται από τους bit κόμβους. Ο πρώτος bit κόμβος λαμβάνει μήνυμα από τον 1^ο και τον 3^ο check κόμβο με τιμή 0. Η πλειοψηφία των μηνυμάτων που δέχεται έχουν διαφορετική τιμή από την αρχική τιμή που διέθετε ο κόμβος οπότε αλλάζει την τιμή του από 1 σε 0. Ο δεύτερος κόμβος συνδέεται με τον 1^ο και τον 2^ο κόμβο με τιμή 1 και από τους δύο. Η τιμή που διέθετε είναι ίδια με αυτή που δέχθηκε οπότε την διατηρεί. Η διαδικασία επαναλαμβάνεται μέχρι να προκύψει τελικά η παρακάτω κωδική λέξη :

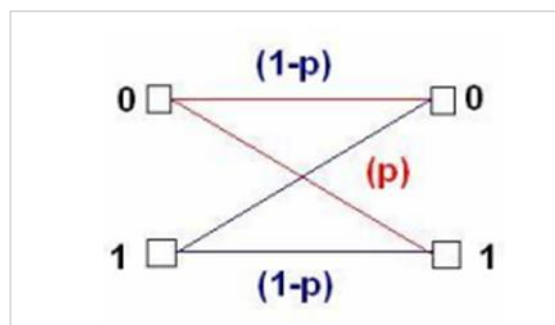
$$M = [0 \ 0 \ 1 \ 0 \ 1 \ 1] .$$

Τελικά ελέγχεται η ισοτιμία για την νέα λέξη που προέκυψε από τους check κόμβους : $L_1 = M_1 \oplus M_2 \oplus M_4 = 0 \oplus 0 \oplus 0 = 0$, $L_2 = M_2 \oplus M_3 \oplus M_5 = 0 \oplus 1 \oplus 1 = 0$, $L_3 = M_1 \oplus$

$M5 \oplus M6 = 0 \oplus 0 \oplus 0 = 0$, $L4 = M3 \oplus M4 \oplus M6 = 1 \oplus 0 \oplus 1 = 0$. Εφόσον, ικανοποιείται η εξίσωση ισοτιμίας οι επαναλήψεις σταματούν και η ανακτημένη λέξη είναι η σωστή.

3.5.3.3 SUM-PRODUCT Αλγόριθμος

Ο αλγόριθμος Sum-Product (SPA) είναι ένας soft decision message passing αλγόριθμος, ο οποίος εισήχθη από τον Gallager, μαζί με τους κώδικες LDPC το 1962. Μοιάζει με τον bit flipping αλλά τα μηνύματα πλέον δεν έχουν τιμή 1 ή 0 ή x. Ο bit flipping έπαιρνε μία απόφαση στην αρχή του αποκωδικοποιητή και κατέληγε στις τιμές των bit, ο sum-product παίρνει μία απόφαση και καταλήγει σε πιθανότητες. Ο αποκωδικοποιητής χρησιμοποιεί κατευθείαν τη λαμβανόμενη ακολουθία r από το κανάλι χωρίς να μεσολαβεί ανιχνευτής. Ο αλγόριθμος είναι γνωστός και ως Belief- Propagation (BPA). Οι πιθανότητες στις οποίες καταλήγει ο αλγόριθμος αφορούν τα εισερχόμενα bits στον αποκωδικοποιητή. Ουσιαστικά, αντιπροσωπεύουν τον BER του καναλιού που είναι άγνωστος στο σύστημα του δέκτη. Είναι εύκολο να υπολογισθούν οι πιθανότητες $p(x=1)$ για μία δυαδική μεταβλητή, δεδομένου ότι $p(x=0)$.



Στο σχήμα παραπάνω φαίνεται η μετάδοση των πιθανοτήτων δεδομένου ότι p είναι η πιθανότητα να έχει μεταδοθεί 0 ενώ έχει σταλεί 1 και αντίστροφα. Η πιθανότητα να έχει πραγματοποιηθεί σωστά η μετάδοση είναι $(1-p)$.

$$L(x) = \log \left(\frac{p(x=0)}{p(x=1)} \right)$$

Η λογαριθμική σχέση χρησιμοποιείται για να εκφράσει τις πιθανότητες και των δύο περιπτώσεων με μία μόνο τιμή, σύμφωνα με τον τύπο παραπάνω. Αν $p(x=0) > p(x=1)$ η τιμή του $L(x)$ είναι θετική και η πιθανότητα να είναι 0 είναι μεγαλύτερη ενώ στην περίπτωση που $p(x=0) < p(x=1)$ η τιμή του $L(x)$ είναι αρνητική και η πιθανότητα να είναι η τιμή του bit 1 είναι μεγαλύτερη. Ο σκοπός του αλγορίθμου είναι να βοηθήσει την μετατροπή του πολλαπλασιασμού σε πρόσθεση. Σκοπός του αλγορίθμου, είναι να ορίσει με τον καλύτερο

τρόπο την εκ των υστέρων πιθανότητα κάθε bit (a posterior), γνωρίζοντας το σήμα που λαμβάνει ο δέκτης, την εξίσωση ελέγχου ισοτιμίας και τα χαρακτηριστικά του διαύλου της διαδρομής από τον πομπό στον δέκτη του καναλιού. Ο αλγόριθμος έχει την ικανότητα να εντοπίζει την καλύτερη λύση για κάθε bit της κωδικής λέξης αλλά όχι την καλύτερη εκτίμηση για συνολικά όλη την κωδική λέξη. Παρακάτω θα ακολουθήσει επεξήγηση των συμβόλων για καλύτερη κατανόηση και ανάγνωση :

Q_{ij}^x : Έκφραση της πιθανότητας ο κόμβος j να βρίσκεται σε κατάσταση x , όπως προκύπτει από την πληροφορία που στέλνουν οι κόμβοι ελέγχου ισοτιμίας που είναι συνδεδεμένοι με τον κόμβο j , εξαιρούνται οι κόμβοι του i .

R_{ij}^x : Έκφραση της πιθανότητας να ικανοποιείται η i -οστή εξίσωση ελέγχου ισοτιμίας δεδομένου ότι ο κόμβος j βρίσκεται στην κατάσταση x .

f_j^x : A priori πιθανότητα το j -οστό bit να είναι x ($x=0$ ή $x=1$). Αρχικά, γίνεται η ενημέρωση των τιμών Q_{ij}^x . Οι πιθανότητες εκφράζονται ανεξαρτήτως από το είδος διαύλου. Έπειτα την αρχικοποίηση, αρχίζει η ανταλλαγή πληροφορίας μεταξύ των bit και check κόμβων. Το σύμβολο R^x παριστάνει την πιθανότητα να ικανοποιείται η i -οστή εξίσωση ελέγχου ισοτιμίας h_i , όπου ο κόμβος j βρίσκεται στην κατάσταση x . Με την συνθήκη ότι ο κόμβος j είναι σε κατάσταση x , η πιθανότητα υπολογίζεται για όλα τα πιθανά αποκωδικοποιημένα διανύσματα d για τα οποία ικανοποιείται η εξίσωση ελέγχου ισοτιμίας. Μετά από τον υπολογισμό των πιθανοτήτων Q και R , σειρά έχει η εκτίμηση της κάθε τιμής d_j bit ξεχωριστά. Η τιμή αυτή, υπολογίζεται στη περίπτωση όπου για x έχουμε 0 ή 1 και με βάση ποια τιμή θα είναι μεγαλύτερη από τις δύο και το j -οστό bit του διανύσματος που προορίζεται για αποκωδικοποίηση. Για να υπολογισθεί το διάνυσμα d θα πρέπει να γίνει η διαδικασία για όλες τις τιμές του j . Αν το διάνυσμα d λαμβάνει έγκυρο αποκωδικοποιημένο μήνυμα τότε η επανάληψη τελειώνει με αποτέλεσμα το διάνυσμα d . Αν δεν συμβεί αυτό, γίνεται επανάληψη χωρίς όμως την αρχικοποίηση μέχρι να βρεθεί έγκυρο αποτέλεσμα ή μέχρι να φτάσει ο αλγόριθμος ένα όριο επαναλήψεων. Τελικά, το διάνυσμα που θα φτάσει στην έξοδο θα είναι λανθασμένο αλλά ο αλγόριθμος θα έχει εκτελέσει την καλύτερη δυνατή πράξη.

3.5.4 Εφαρμογές

Το 2003, ένας τύπος κώδικας LDPC, ο IRA (irregular repeat accumulate) ξεπέρασε τους κωδικούς Turbo και έτσι επιλέχθηκε για κωδικός διόρθωσης σφαλμάτων στο νέο πρότυπο DVB-S2 (Digital Video Broadcasting–Satellite 2) για την ψηφιακή τηλεόραση. Τα πρότυπα

DVB-S2 (Digital Video Broadcasting-Satellite-Second Generation) , DVB-C2(Digital Video Broadcasting—Cable Second Generation) και DVB-T2 (Digital Video Broadcasting—Second Generation Terrestrial), χρησιμοποιούν έναν κώδικα BCH (Bose–Chaudhuri–Hocquenghem) για την εξάλειψη των σφαλμάτων που έχουν μείνει μετά την αποκωδικοποίηση LDPC.

Το 2008, οι LDPC ξεπέρασαν τους συνελκτικούς κώδικες Turbo λόγω της χαμηλότερης πολυπλοκότητας στην διαδικασία της αποκωδικοποίησης. Σε αυτό συνέβαλε και το μειονέκτημα των Turbo κωδικών να παρουσιάζουν μεγάλο επίπεδο σφάλματος στο επιθυμητό εύρος λειτουργίας. Το πρότυπο που χρησιμοποιείται κυρίως είναι το Forward Error Correction το οποίο βασίζεται σε μία συγκεκριμένη κατηγορία των LDPC κωδικών. Ειδικότερα ο πίνακας H έχει την μορφή $H = [H_1|H_2]$, ο H_1 πίνακας είναι αραιός και ψευδοτυχαίος ικανοποιώντας κάποιους περιοδικούς περιορισμούς και ο H_2 πίνακας είναι αραιός, κάτω τριγωνικός πίνακας. Χρησιμοποιείται στο πρότυπο G.hn/G.9960 κωδικών (Next-Generation Wired Home Networking). Το πρότυπο αυτό προσφέρει υπηρεσίες δικτύων χρησιμοποιώντας τηλεφωνικές γραμμές, γραμμές ισχύος και ομοαξονικά καλώδια. Μερικά συστήματα OFDM (orthogonal frequency-division multiplexing), δηλαδή συστήματα με ορθογώνια πολυπλεξία διαίρεσης συχνότητας προσθέτουν μία επιπλέον διόρθωση εξωτερικών σφαλμάτων που διορθώνει τα περιστασιακά σφάλματα, αυτό ονομάζεται το κατώτατο όριο των σφαλμάτων, που διαπερνούν τον εσωτερικό κώδικα διόρθωσης ακόμη και αν πρόκειται για χαμηλούς ρυθμούς σφάλματος bit.

Χρησιμοποιούνται επίσης στο πρότυπο 10GBase-T Ethernet (IEEE 802.3an) το οποίο στέλνει δεδομένα με ταχύτητα 10Gbit/sec σε ζεύγος συνεστραμμένων καλωδίων. Εδώ οι κώδικες LDPC είναι βασισμένοι σε κώδικες Reed-Solomon έτσι ώστε να καθοριστεί ο πίνακας γεννήτορα.

Οι κώδικες LDPC έχουν εφαρμογή και στο πρότυπο WiMAX. Το πρότυπο WiMAX, δηλαδή το πρότυπο IEEE 802.16e για μικροκυματικές επικοινωνίες, είναι βασισμένο στην χρήση των LDPC κωδικών.

3.5.5 Πλεονεκτήματα -Μειονεκτήματα LDPC

Οι κώδικες LDPC είναι ιδανικοί για προσεγγίσεις απόδοσης ποσοστού σφάλματος σύμφωνα με το όριο Shannon. Αποτελούν κατηγορία γραμμικών μπλοκ κωδικών των οποίων ο έλεγχος ισοτιμίας έχει χαμηλή πυκνότητα (low density), δηλαδή είναι αραιοί. Η κανονικότητα (regular code) σε συνδυασμό με την δομή του κώδικα καθιστούν την κωδικοποίηση πιο αποτελεσματική και πιο απλοποιημένη την αποκωδικοποίηση. Η μεγάλη περιφέρεια (girth)

επιταχύνει την αποκωδικοποίηση και βελτιώνει την απόδοση σε μεγάλη κλίμακα SNR, επιβραδύνοντας την έναρξη του σφάλματος (low floor error). Τα μήκη των μπλοκ πληροφοριών μπορούν να είναι αρκετά μεγάλα στους LDPC κώδικες.

Για να εξαχθούν ικανοποιητικά στοιχεία απαιτούνται μεγάλα μεγέθη πινάκων H . Η κλιμακούμενη αρχιτεκτονική των κωδίκων LDPC επιβάλλουν περιορισμούς στην μήτρα του κώδικα. Αρκετές φορές για εκτιμήσεις απαιτούνται μεγάλοι πόροι σε υλικά (hardware), γεγονός που καθιστά την χρήση τους δεσμευτική. Ξοδεύεται αρκετός χρόνος προκειμένου να βρεθεί η βέλτιστη λύση. Μέρη τα οποία είναι μερικώς ή ολικώς ημι-παράλληλα πρέπει να αντιμετωπίσουν τα θέματα σύγκρουσης στην μνήμη, την διευθυνσιοδότηση και τον σύνθετο έλεγχο.

3.6 AFF3CT

Σύμφωνα με τις ιστοσελίδες <https://aff3ct.readthedocs.io/>, <https://www.researchgate.net/> και <https://aff3ct.github.io/> δύνανται να ειπωθούν οι παρακάτω αναφορές σχετικά με τον κώδικα AFF3CT.

Το AFF3CT είναι ένας προσομοιωτής που εστιάζει στο επίπεδο κωδικοποίησης καναλιού. Χρησιμοποιείται για την αναπαράσταση των BER/FER σε συνδυασμό με το SNR(signal-to-noise ratio). Ο προσομοιωτής βασίζεται σε μία μέθοδο Monte Carlo, δηλαδή ο πομπός εκπέμπει καρέ που προκαλούν τυχαίο θόρυβο από το κανάλι προς τον δέκτη, ο οποίος προσπαθεί να αποκωδικοποιήσει τα καρέ με θόρυβο. Ο πομπός εκπέμπει καρέ μέχρι να επιτευχθεί σταθερός αριθμός σφαλμάτων καρέ. Όταν η τιμή του SNR μειώνεται, αυξάνεται ο αριθμός των πλαισίων και ο χρόνος προσομοίωσης.

3.6.1 OUTPUT

Η έξοδος ενός προσομοιωτή θα είναι η εξής:

simulation is running...

Bit Error Rate (BER) and Frame Error Rate (FER) depending on the Signal Noise Ratio (SNR)							Global throughput and elapsed time	
Es/No (dB)	Eb/No (dB)	FRA	BE	FE	BER	FER	SIM_THR (Mb/s)	ET/RT (hhmmss)
0.25	1.00	532	84173	532	9.18e-02	1.00e+00	6.28	00h00'00
0.50	1.25	536	80920	536	8.76e-02	1.00e+00	8.80	00h00'00
0.75	1.50	535	75875	535	8.23e-02	1.00e+00	8.72	00h00'00
1.00	1.75	536	71227	536	7.71e-02	1.00e+00	9.06	00h00'00
1.25	2.00	536	65426	536	7.08e-02	1.00e+00	10.02	00h00'00
1.50	2.25	535	57336	535	6.22e-02	1.00e+00	10.98	00h00'00
1.75	2.50	545	48081	536	5.12e-02	9.83e-01	11.21	00h00'00
2.00	2.75	580	36856	534	3.69e-02	9.21e-01	11.96	00h00'00
2.25	3.00	752	27246	522	2.10e-02	6.94e-01	15.61	00h00'00
2.50	3.25	1179	22251	529	1.10e-02	4.49e-01	24.37	00h00'00
2.75	3.50	2762	16320	514	3.43e-03	1.86e-01	55.40	00h00'00
3.00	3.75	6756	14102	504	1.21e-03	7.46e-02	126.52	00h00'00
3.25	4.00	23243	13960	501	3.49e-04	2.16e-02	335.54	00h00'00
3.50	4.25	101560	10960	500	6.26e-05	4.92e-03	732.08	00h00'00
3.75	4.50	447510	11476	500	1.49e-05	1.12e-03	989.43	00h00'00

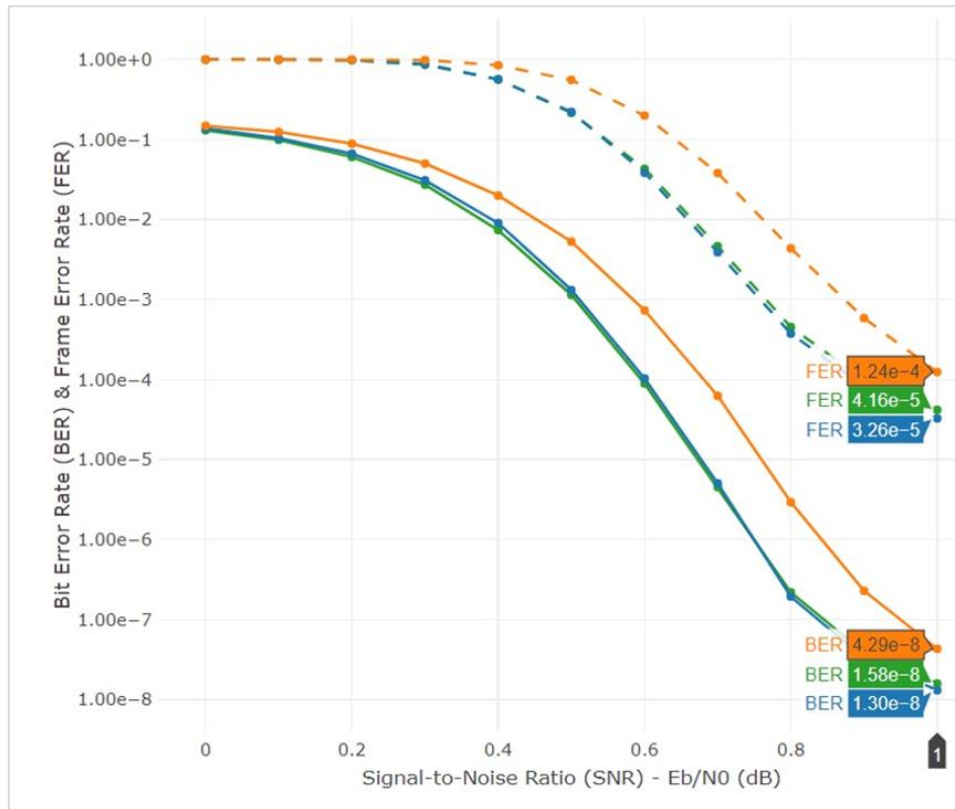
of the simulation.

Κάθε γραμμή ξεκινάει από τον χαρακτήρα # και αποτελείται από τις ακόλουθες παραμέτρους:

- BER: το ποσοστό σφάλματος bit
- FER: το ποσοστό σφάλματος καρέ
- BE: αριθμός σφαλμάτων bit
- FE: αριθμός σφαλμάτων πλαισίου
- FRA: αριθμός προσομοιωμένων πλαισίων
- Eb/No: Αναλογία φασματικής πυκνότητας ενέργειας ανά bit προς θόρυβο ισχύος
- Es/No: SNR του σήματος στον δέκτη, δηλαδή το σηματοθορυβικό κλάσμα που δείχνει πόσο ισχυρότερο είναι το σήμα λήψης από το θόρυβο του δέκτη.
- SIM_THR: Η απόδοση της προσομοίωσης
- ET/RT: Εμφανίζει μία εκτίμηση του υπολειπόμενου χρόνου (RT) εφόσον ολοκληρωθούν οι υπολογισμοί, είναι ο συνολικός χρόνος που έχει παρέλθει (ET).

3.6.2 TURBO Προσομοίωση

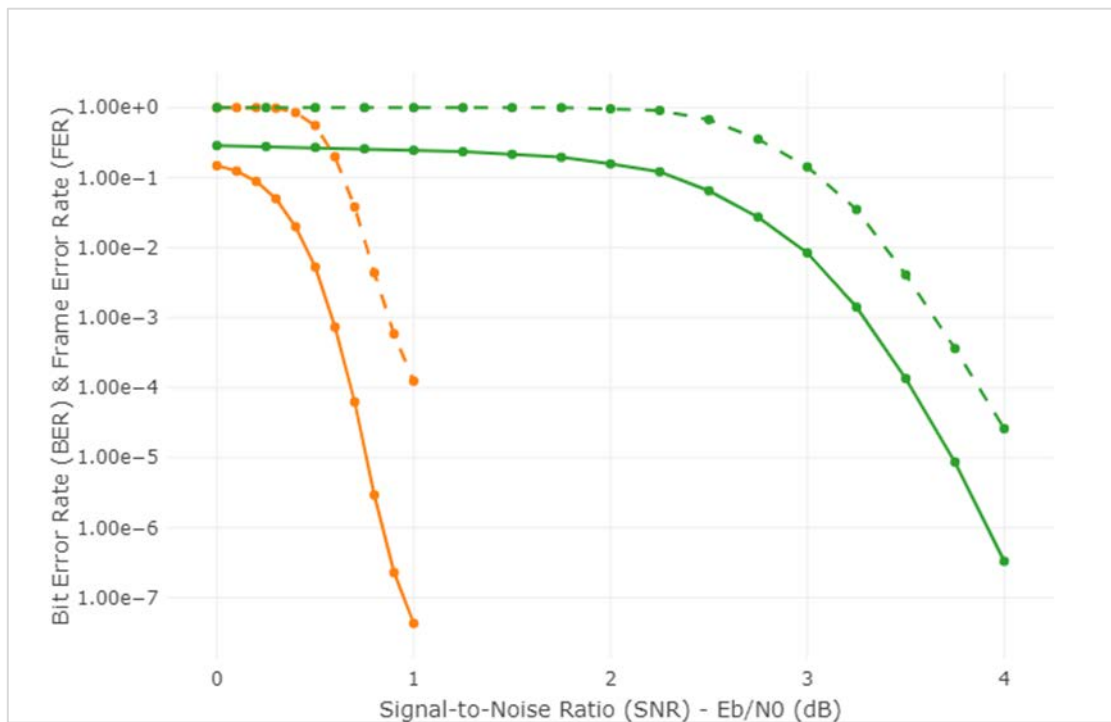
3.6.2.1 Απόδοση BER/FER για FRAME SIZE N=18444



Σχήμα 35: Turbo (18444.1644) και $i=11$. Πηγή: <https://aff3ct.github.io/>

Γενικά, όσο αυξάνεται το μέγεθος του πλαισίου N τόσο καλύτερη απόδοση θα έχει ο Turbo κώδικας. Για να διατηρηθεί η προσομοίωση, ο αριθμός των πλαισίων για κάθε SNR έχει υπολογιστεί ως 500. Άρα, για 18444, 1644 bits στάλθηκαν σε κάθε τιμή SNR για να αποδειχθεί το ποσοστό σφάλματος (BER) και ο ρυθμός σφαλμάτων πλαισίου (FER). Το εύρος θορύβου SNR που χρησιμοποιήθηκε ήταν από 0 έως 1 dB, σε κανάλι AWGN. Ο αριθμός των επαναλήψεων είναι $i=11$ και ο ρυθμός του κώδικα είναι $R=1/3$. Στο συγκεκριμένο παράδειγμα, αποδεικνύεται ότι όσα περισσότερα bit χρησιμοποιήσει ο κώδικας στον αλγόριθμο BCJR τόσο καλύτερη απόδοση έχει το BER/FER. Η πορτοκαλί καμπύλη αποτελείται από 8 bit, η πράσινη από 16 bit και η μπλε από 32 bit. Συμπερασματικά λοιπόν, η μπλε έχει την καλύτερη απόδοση από όλους τους άλλους όπως φαίνεται και στην παραπάνω εικόνα.

3.6.2.2 Απόδοση BER/FER με διαφορετικό FRAME SIZE N

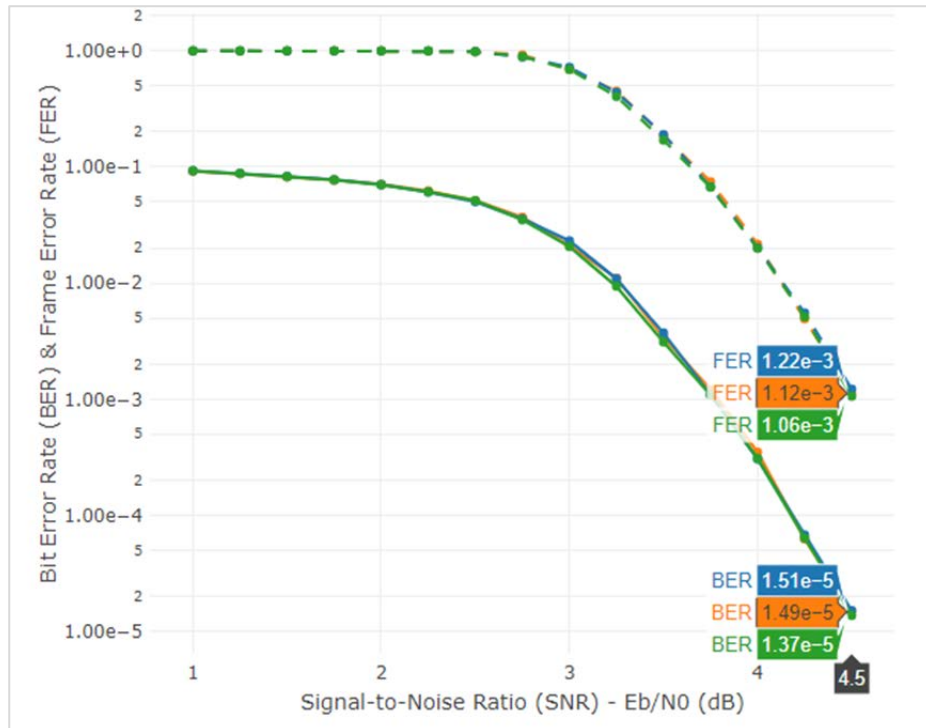


Σχήμα 36: Σύγκριση επιδόσεων κώδικα Turbo για N=18444 και N=3084. Πηγή: <https://aff3ct.github.io/>

Η σύγκριση των επιδόσεων του Turbo κώδικα μπορεί να γίνει για διαφορετικά frame sizes N και για $R=1/3$. Αυτό που αποδεικνύεται πάλι, είναι ότι με την αύξηση του N, η απόδοση του BER/FER βελτιώνεται. Καλύτερη απόδοση BER/FER έχει η πορτοκαλί καμπύλη για το μέγεθος πλαισίου N 18444 όπου επιτυγχάνεται σε SNR 1 dB και $i=11$ επαναλήψεις και κανάλι AWGN, ενώ η πράσινη καμπύλη για το μέγεθος πλαισίου N 3084 όπου επιτυγχάνεται σε SNR 4 dB για $i=17$ επαναλήψεις και κανάλι AWGN. Επίσης, αποδίδει καλύτερα η πορτοκαλί από ό,τι η πράσινη με λιγότερες επαναλήψεις και με κέρδος κωδικοποίησης 3 dB. Μεγαλύτερα frame sizes σημαίνει μεγαλύτερη καθυστέρηση, εφόσον η κωδικοποίηση και η αποκωδικοποίηση γίνεται ανά frame. Έτσι, η βελτίωση απόδοσης επιτυγχάνεται με το κόστος της αυξημένης λανθάνουσας κατάστασης.

3.6.3 POLAR Προσομοίωση

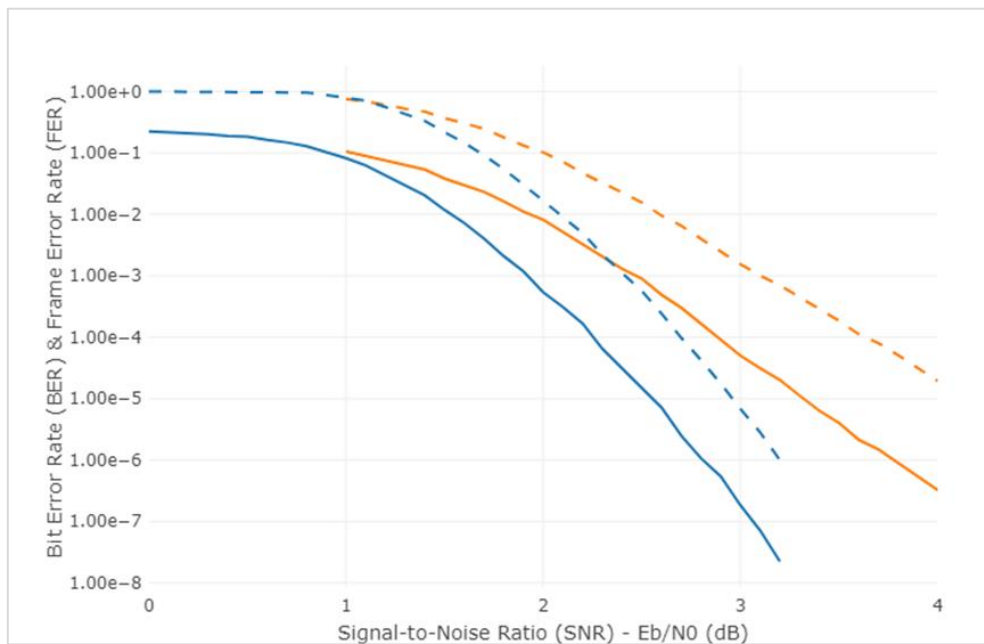
3.6.3.1 Απόδοση BER/FER για FRAME SIZE N=2048



Σχήμα 37: Polar (2048.1723) για $i=15$. Πηγή: <https://aff3ct.github.io/>

Παρατηρώντας την παραπάνω προσομοίωση καταλήγουμε στο συμπέρασμα ότι καθώς αυξάνεται η τιμή του πλαισίου N , επιτυγχάνεται καλύτερη τιμή απόδοσης στον κώδικα. Η συχνότητα του κώδικα έχει την τιμή 500. Συνεπώς, για 2048 τιμή πλαισίου και 1723 bits στάλθηκαν σε κάθε τιμή SNR για να επιτευχθούν τα ποσοστά σφάλματος BER και FER. Οι επαναλήψεις που πραγματοποιούνται είναι 15 και ο ρυθμός του κώδικα είναι $R=4/5$. Στο συγκεκριμένο παράδειγμα, η καλύτερη απόδοση του BER/FER αντιστοιχεί με τον αλγόριθμο SC για 32 bit, η επόμενη καλύτερη για 8 bit και μετά για 16. Η πράσινη καμπύλη αφορά τον κώδικα με τα 32 bit, η πορτοκαλί με τα 8 και η μπλε με τα 16. Το εύρος θορύβου ξεκινά από 1 έως 4,5 dB.

3.6.3.2 Απόδοση BER/FER με διαφορετικό FRAME SIZE N

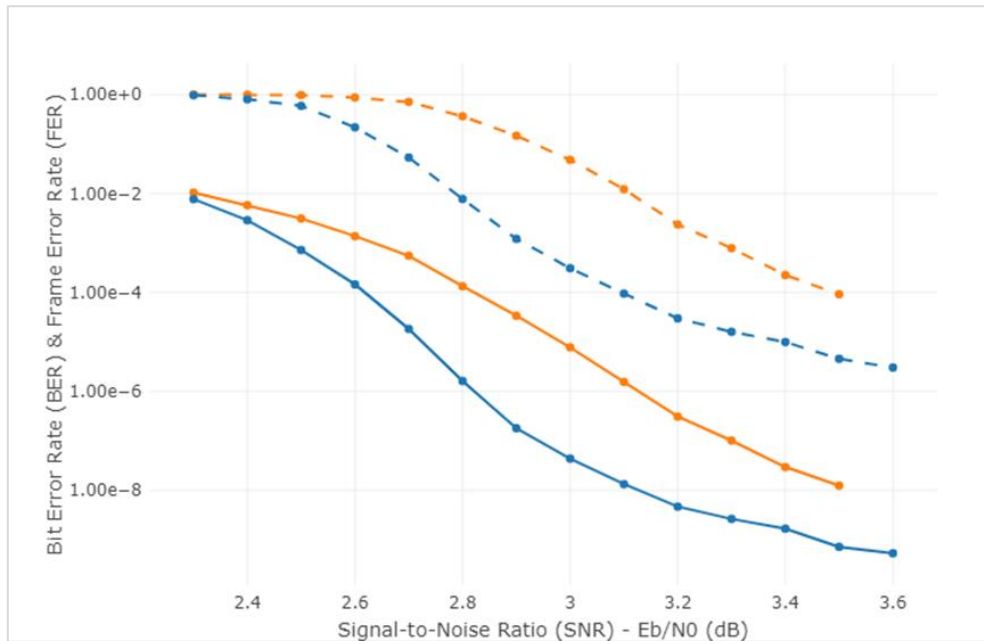


Σχήμα 38: Σύγκριση επιδόσεων κώδικα Polar για N=1024 και N=4096. Πηγή: <https://aff3ct.github.io/>

Στο παραπάνω σχήμα φαίνονται οι επιδόσεις για διαφορετικό μέγεθος μπλοκ και ίδιο ρυθμό. Στην πορτοκαλί καμπύλη, απεικονίζεται ο κώδικας με αριθμό πλαισίου N= 1024, 2048 bit πληροφορίας που στάλθηκαν σε κάθε τιμή SNR και έχει εφαρμογή σε δίκτυα 5ης γενιάς με 32 bit και στην μπλε καμπύλη απεικονίζεται ο κώδικας με αριθμό πλαισίου N=4096, 512 bit πληροφορίας με επίσης 32 bit. Το κέρδος της κωδικοποίησης είναι $4-3,2= 0,8$. Η εξαγωγή (output) του κώδικα με μπλε χρώμα και μεγαλύτερο μέγεθος μπλοκ είναι καλύτερη από το αποτέλεσμα του άλλου κώδικα. Αυτό συμβαίνει διότι με μικρότερη κατανάλωση ισχύος μπορεί να επιτύχει μικρότερη τιμή BER/FER. Από αυτήν την σύγκριση, παρατηρείται ότι μεγάλα μεγέθη μπλοκ απαιτούν μικρότερο Eb/No προκειμένου να επιτευχθεί συγκεκριμένη τιμή BER/FER. Συνεπώς, επιβεβαιώνονται οι θεωρητικές έρευνες που αναφέρουν ότι για ανάγκη μεγάλου μπλοκ μεγέθους επιτυγχάνεται χαμηλή τιμή BER/FER. Είναι προφανές ότι το αποτέλεσμα σχετίζεται με την ταχύτητα της πόλωσης. Όταν πρόκειται για μικρά μεγέθη μπλοκ αναμένεται όχι τόσο ικανοποιητική πόλωση καναλιού και είναι εμφανές και στο σχήμα παραπάνω. Αυτό επιβεβαιώνει την καθολικότητα των Polar κωδικών και την εξάρτηση που έχουν από τον δίαυλο επικοινωνίας.

3.6.4 LDPC Προσομοίωση

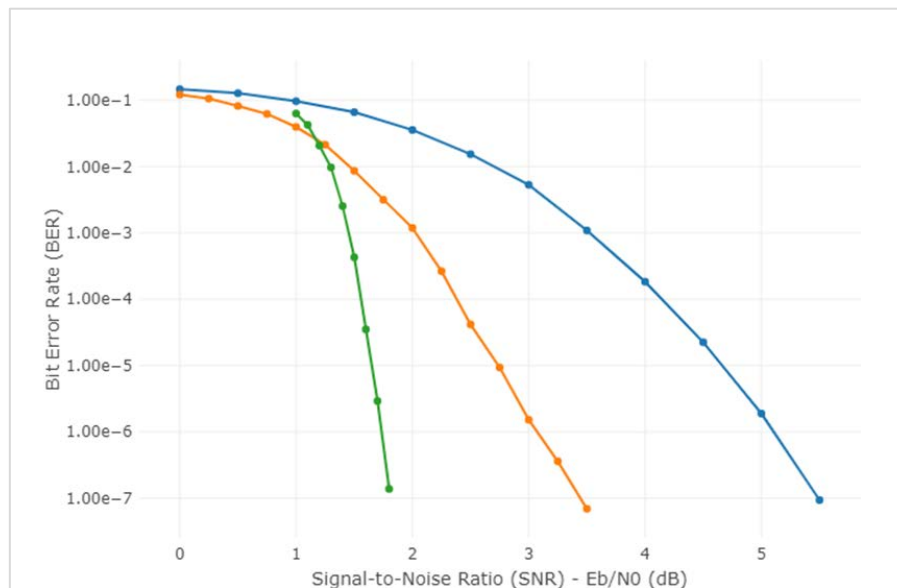
3.6.4.1 Απόδοση BER/FER για (16200,11880) με Διαφορετικούς Αλγόριθμους



Σχήμα 39: Απόδοση αλγορίθμων NMS και SPA για LDPC(16200,11880). Πηγή: <https://aff3ct.github.io/>

Στη συγκεκριμένη προσομοίωση LDPC κώδικα, απεικονίζονται δύο διαφορετικοί τύποι αλγορίθμων για 16200 αριθμό πλαισίου, 11880 bits τα οποία στάλθηκαν σε κάθε τιμή SNR για να αποδειχθεί το ποσοστό σφάλματος (BER) και ο αριθμός σφαλμάτων πλαισίου (FER). Το εύρος θορύβου SNR που χρησιμοποιήθηκε ήταν από 2,3 έως 3,6. Συγκεκριμένα, η πορτοκαλί καμπύλη απεικονίζει τον αλγόριθμο SPA με αριθμό επαναλήψεων $i=13$ και ρυθμό κώδικα $R=11/15$ και η μπλε καμπύλη τον NMS με $i=14$ και $R=11/15$. Παρατηρείται, ότι ο κώδικας NMS για μεγαλύτερη τιμή του SNR έχει μικρότερη τιμή BER, με περισσότερες επαναλήψεις, σε αντίθεση με τον SPA, ο οποίος για μικρότερη τιμή SNR διαθέτει μεγαλύτερη τιμή BER.

3.6.4.2 Απόδοση BER/FER για διαφορετικό FRAME SIZE N



Σχήμα 40: K 128,576,8000. Πηγή: <https://aff3ct.github.io/>

Στην παραπάνω εικόνα, αυτό που απεικονίζεται είναι ένας LDPC κώδικας με διαφορετικά N και K αλλά με ρυθμό $R=1/2$. Σημαντικό ρόλο στην προσομοίωση είναι το frame size. Η πράσινη καμπύλη λοιπόν, απεικονίζει τον LDPC(8000.4000) SPA 20ite MacKay, ο οποίος είναι ο καλύτερος κώδικας που αποδίδει στο συγκεκριμένο παράδειγμα. Το εύρος ζώνης ξεκινά από 1 έως 1,8 dB, με $i=9$ επαναλήψεις και θα τερματίσει με $BER=1,38e-7$. Η πορτοκαλί καμπύλη απεικονίζει τον LDPC(576.288) SPA 100ite WiMax, ο οποίος είναι ο δεύτερος καλύτερος κώδικας στη γραφική παράσταση. Το εύρος ζώνης ξεκινά από 0 έως 3,5 dB, με $i=15$ επαναλήψεις και θα τερματίσει με $BER=6,93e-8$. Τέλος, είναι ο LDPC(128.64) SPA 50ite CCSDS ο οποίος έχει εύρος ζώνης από 0 έως 5,5 dB και θα τερματίσει με $BER=9,37e-8$. Ανάμεσα στην πορτοκαλί και πράσινη καμπύλη υπάρχει κέρδος κωδικοποίησης 1,7 dB, στην μπλε και πορτοκαλί καμπύλη υπάρχει κέρδος κωδικοποίησης 2 dB και ανάμεσα στην μπλε και πράσινη καμπύλη υπάρχει κέρδος κωδικοποίησης 3,7 dB. Συμπερασματικά, αυτό που αποδεικνύει το συγκεκριμένο παράδειγμα είναι, καθώς το μήκος N αυξάνει, οι επιδόσεις βελτιώνονται. Η απόσταση μεταξύ των καμπυλών αυξάνει σύμφωνα με την αύξηση του E_b/N_0 .

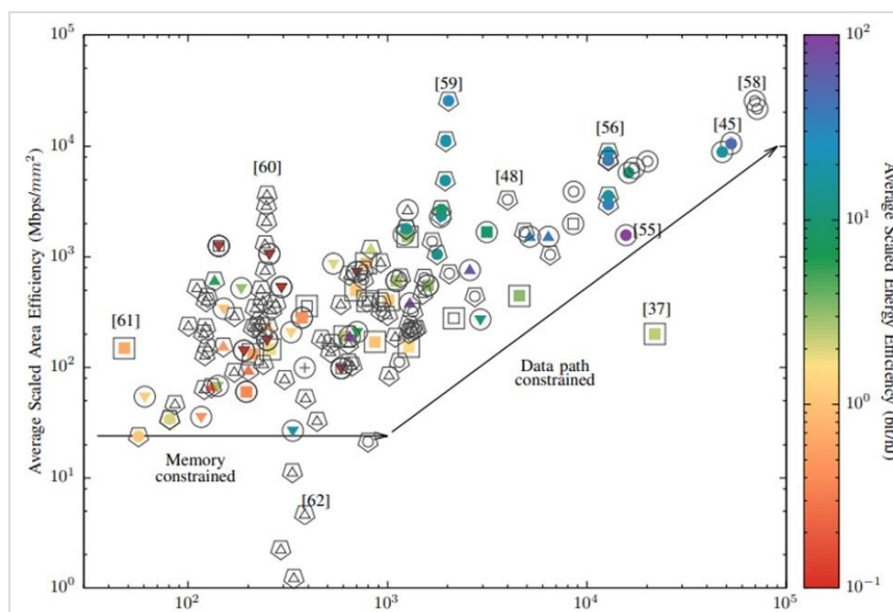
3.7 Συγκρίσεις Τεχνικών Κωδικοποίησης

3.7.1 TURBO vs POLAR vs LDPC

Βασική διαφορά των τριών κωδικών είναι ότι οι κώδικες Polar ανακαλύφθηκαν το 2008 από τον Erdal Arıkan με προηγμένες τεχνικές διόρθωσης σφαλμάτων ενώ οι κώδικες Turbo ανακαλύφθηκαν γύρω στο 1990-1991 αλλά δημοσιεύθηκαν για πρώτη φορά το 1993 από τους

Cladue Berrou, Alain Glavieux και Punya Thitimajshima. Οι LDPC κώδικες αναπτύχθηκαν για πρώτη φορά από τον Gallager το 1963 αλλά ξεχάστηκαν μέχρι που ανακαλύφθηκαν εκ νέου το 1996. Οι κώδικες Polar χρησιμοποιούνται κατά κόρον σε τεχνικές κωδικοποίησης καναλιού σε δίκτυα 5^{ης} γενιάς και για αυτό θεωρείται ένα από τα μεγαλύτερα επιτεύγματα της εποχής. Οι κώδικες Turbo χρησιμοποιούνται σε κινητές επικοινωνίες 3G/4G (σε συστήματα LTE, UMTS και σε δορυφόρους επικοινωνιών) ενώ, οι κώδικες LDPC χρησιμοποιούνται για δίκτυα 5G, για δορυφορικές επικοινωνίες καθώς και για το Deep Space Network. Όσον αφορά την πολυπλοκότητα, οι κώδικες LDPC παρουσιάζουν μέτρια πολυπλοκότητα $O(n^2)$ αλλά με τη βοήθεια του πίνακα H και του χαρακτηριστικού της αραιότητας μπορεί να επιτύχει πολυπλοκότητα $O(n)$. Οι κώδικες Polar, έχουν μέτρια τιμή πολυπλοκότητας κωδικοποίησης και αποκωδικοποίησης $O(N \log N)$. Οι κώδικες Turbo από την άλλη, παρουσιάζουν μεγάλη τιμή πολυπλοκότητας διότι στην διαδικασία της αποκωδικοποίησης χρησιμοποιούν τον αλγόριθμο MAP. Με αποτέλεσμα, ανάλογα με την πολυπλοκότητα του LDPC κώδικα $O(n^2)$ ή $O(n)$ και την πολυπλοκότητα των κωδικών Polar επιλέγεται ο καλύτερος. Η απόδοση διόρθωσης σφαλμάτων των κωδικών Polar βελτιώνεται πιο αργά με το μήκος block σε αντίθεση με τους κώδικες Turbo και LDPC, όμως μπορούν να επιτύχουν εξαιρετική απόδοση σφαλμάτων σε συνδυασμό με χαμηλή πολυπλοκότητα αποκωδικοποίησης όταν χρησιμοποιούνται με πιο προηγμένους αλγορίθμους κωδικοποίησης. Η αποκωδικοποίηση BP (Belief Propagation Decoding) και SC (Successive Cancellation Decoding) έχουν κατώτερη διόρθωση σφαλμάτων σε σύγκριση με τους κώδικες LDPC και Turbo. Οι κώδικες αποκωδικοποίησης LDPC και Turbo έχουν υψηλότερη απόδοση υλικού (hardware), έναντι των κωδικών Polar. Οι κώδικες Polar εμφανίζουν ορισμένους περιορισμούς όταν χρησιμοποιούνται σε βιομηχανικές εφαρμογές. Ο σχεδιασμός τους επιτυγχάνει χωρητικότητα όταν μεγάλα μεγέθη block είναι ασυμπτωματικά μεγάλα μαζί με διαδοχικό αποκωδικοποιητή ακύρωσης. Όμως, με τα μεγέθη block που χρησιμοποιεί η βιομηχανία οι κώδικες Polar έχουν κακή απόδοση διαδοχικής ακύρωσης σε αντίθεση με τους κώδικες LDPC και Turbo, οι οποίοι έχουν καλά καθορισμένα και εφαρμοσμένα σχήματα κωδικοποίησης. Το 2008, οι LDPC ξεπέρασαν τους κώδικες Turbo στο σύστημα διόρθωσης σφαλμάτων FEC για το πρότυπο ITU-T G.hn. Ο ρυθμός κώδικα $R=k/n$ που χρησιμοποιούν οι κώδικες Polar είναι συνήθως 1/2 και πιο σπάνια 3/4, 7/8. Για τους κώδικες Turbo είναι συνήθως 1/2 και πιο σπάνια 1/3 και για τους κώδικες LDPC είναι συνήθως 1/2 και πιο σπάνια 4/6 και 5/7. Επιπρόσθετα, η κωδικοποίηση και η αποκωδικοποίηση τους είναι εντελώς διαφορετική στο κάθε ένα. Συγκρίνοντας, τον αριθμό επαναλήψεων των τριών κωδικών γίνεται κατανοητό ότι η τιμή των επαναλήψεων δε παίζει κάποιο σημαντικό ρόλο στους κώδικες Polar. Για τους LDPC η αποκωδικοποίηση

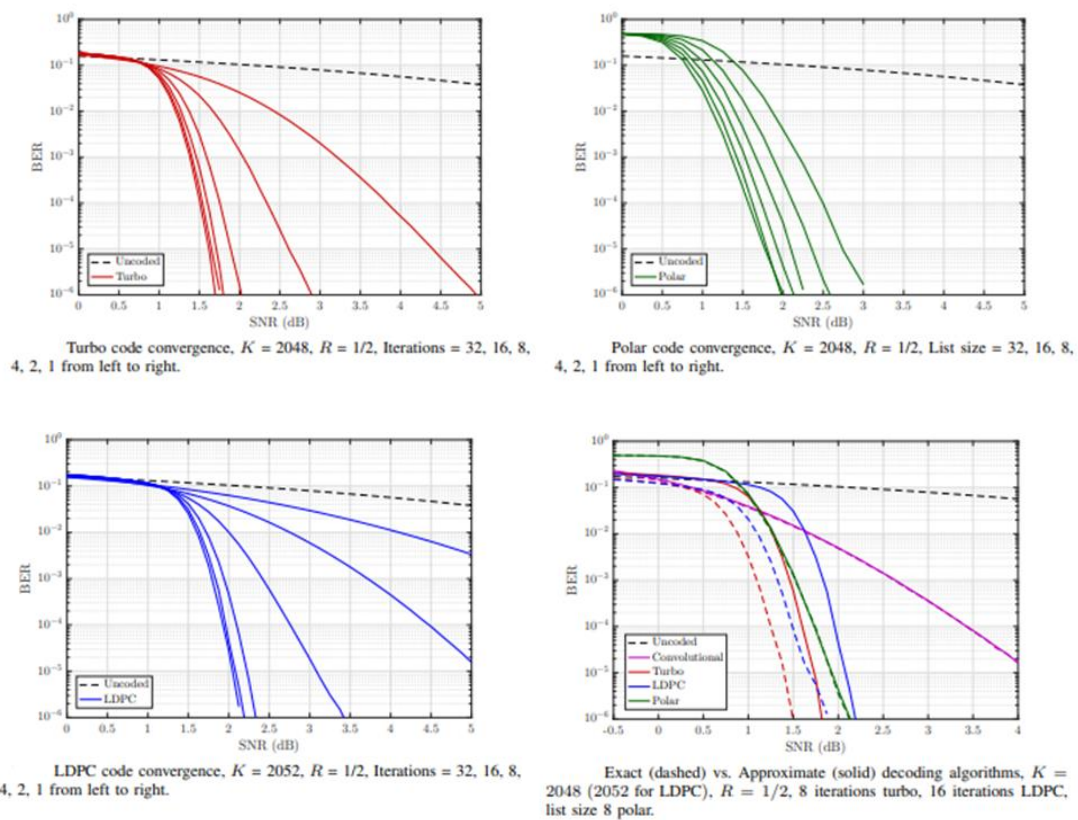
συγκλίνει μεταξύ των 15 και 20 επαναλήψεων ενώ για τους Turbo είναι γνωστό ότι 10 επαναλήψεις επαρκούν για τη σύγκλιση. Επομένως, για μία διαρκεί σύγκριση μεταξύ LDPC και Turbo ο αριθμός των επαναλήψεων επιλέγεται να είναι 20 και 10 αντίστοιχα για τη σωστή εξαγωγή αποτελεσμάτων. Σχετικά με τις διαφορές στην αποκωδικοποίηση, οι Turbo έχουν χαμηλή απόδοση πληροφοριών. Η απόδοση αυξάνεται όταν υπάρχει πλήρης παράλληλη δομή. Για τους LDPC, η απόδοση είναι υψηλή για υψηλούς ρυθμούς και για παράλληλη δομή, ενώ για τους Polar η απόδοση είναι υψηλή για υψηλή τιμή pipeline(σύνολο στοιχείων επεξεργασίας δεδομένων σε σειρά). Για την ενεργειακή απόδοση, οι Turbo έχουν χαμηλή αποτελεσματικότητα για τους περισσότερους ρυθμούς κώδικα αλλά για τους χαμηλούς ρυθμούς εμφανίζουν υψηλή αποτελεσματικότητα. Οι LDPC, έχουν υψηλή για τους περισσότερους ρυθμούς κώδικα αλλά χαμηλή για χαμηλούς ρυθμούς και οι Polar υψηλή για τους περισσότερους ρυθμούς κώδικα. Μεγαλύτερη ευελιξία, διαθέτουν οι Polar έπειτα ακολουθούν οι Turbo και με μέτρια οι LDPC.



Σχήμα 41: Απόδοση περιοχής και ενέργειας στους διάφορους αποκωδικοποιητές καναλιών. Πηγή: <https://aff3ct.github.io/>

Στο παραπάνω σχήμα, απεικονίζονται οι διαφορές των αποκωδικοποιητών των τριών κωδικών σχετικά με τη ροή της πληροφορίας. Οι υλοποιήσεις του Turbo αποκωδικοποιητή φαίνονται να έχουν το μεγαλύτερο τμήμα που σχετίζεται με χαμηλή ροή πληροφοριών που μπορούν να αποδοθούν σε χαμηλούς βαθμούς παραλληλισμού οι οποίοι επιβάλλονται από εξαρτήσεις δεδομένων συγκεκριμένων αλγορίθμων. Από την άλλη, οι υλοποιήσεις του Polar αποκωδικοποιητή είναι περιορισμένες εξαιτίας των αυστηρών εξαρτήσεων των αλγορίθμων SC και SCL. Οι LDPC υλοποιήσεις αποκωδικοποιητή έχουν το πιο υψηλό ποσοστό το οποίο

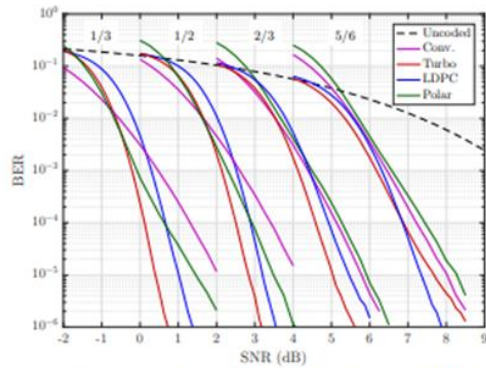
σχετίζεται με υψηλές αποδόσεις πληροφοριών λόγω του υψηλού βαθμού παραλληλισμού που προκύπτει από τον αλγόριθμο min-sum. Βασική απαίτηση, για επικοινωνία των δικτύων 5G είναι η επίτευξη μέγιστης τιμής απόδοσης πληροφοριών. Όπως λοιπόν φαίνεται από το σχήμα, ο αποκωδικοποιητής Turbo καταφέρνει τις πληροφορίες των 20 Gbps. Οι μοναδικοί LDPC αποκωδικοποιητές που έχουν επιτύχει την ταχύτητα των 20 Gbps έχουν υιοθετήσει πλήρως παράλληλες αρχιτεκτονικές. Η εφαρμογή LDPC ASIC (Application specific integrated circuit) πέρα από την υψηλή απόδοση πληροφοριών που επιτυγχάνει, καταφέρνει και εξαιρετική ενεργειακή απόδοση.



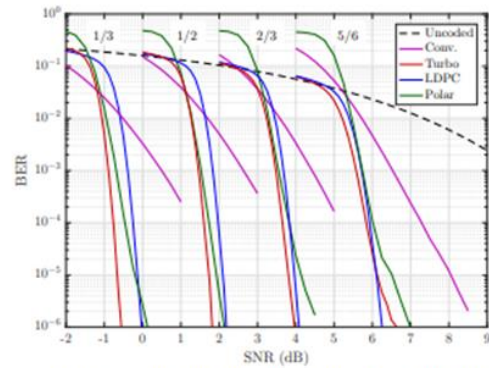
Σχήμα 42: Τρόπος με τον οποίο επηρεάζεται η απόδοση σε συνάρτηση με τον αριθμό των επαναλήψεων για Turbo και LDPC και για το μέγεθος λίστας του Polar. Πηγή: <https://aff3ct.github.io/>

Παραπάνω, εξετάζεται ο τρόπος με τον οποίο επηρεάζεται η απόδοση σε συνάρτηση με τον αριθμό των επαναλήψεων για Turbo και LDPC και για το μέγεθος λίστας του Polar. Χρησιμοποιούνται οι αλγόριθμοι αποκωδικοποίησης MaxLog-MAP για τον Turbo, Layered Min-Sum για τον LDPC και List-SC για τον Polar. Τα αποτελέσματα αφορούν ρυθμό $R=1/2$, μήκος block $K=2048$ για τους Turbo και Polar και $K=2052$ για τον LDPC. Από τις εξαγωγές (output) προκύπτει ότι ο κώδικας Turbo πραγματοποιεί 8 επαναλήψεις με όχι τόσο μεγάλο κέρδος. Για 16 επαναλήψεις του LDPC υπάρχει επαρκές κέρδος ενώ για 32 επαναλήψεις ελάχιστο κέρδος. Σχετικά με τον Polar κώδικα καλύτερη απόδοση επιτυγχάνετε για

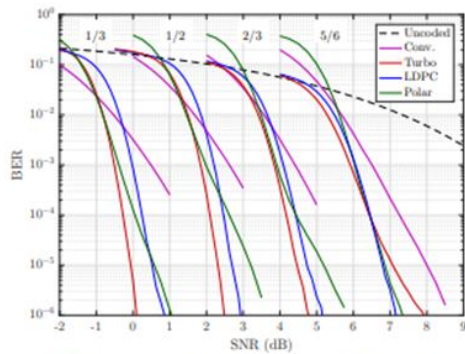
μεγαλύτερα μεγέθη λιστών και για τη λίστα μεγέθους 32 παρουσιάζεται ο περιορισμός στην περιοχή χαμηλής τιμής BER. Για τις άλλες εξαγωγές (output) επιλέγονται 8 επαναλήψεις για τον Turbo, 16 για τον LDPC και μέγεθος λίστας για τον Polar. Στην τελευταία εξαγωγή, φαίνεται η ακριβής απόδοση των αλγορίθμων. Για τον Polar, η διαφορά είναι σχεδόν ανύπαρκτη όμως για τον Turbo και LDPC υπάρχει σημαντική διαφορά της τάξης των 0,37 έως 0,47 dB.



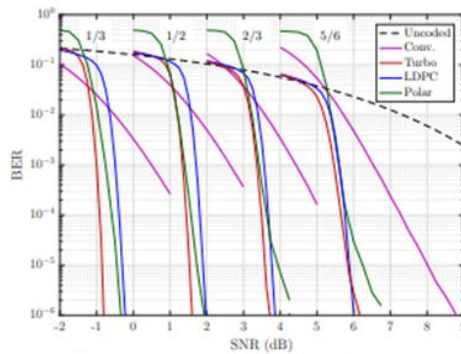
BER comparison for different code rates, $K = 256$ (For LDPC, $K = 252$ for $R = 1/2$, and $1/3$, and $K = 260$ for $R = 5/6$.)



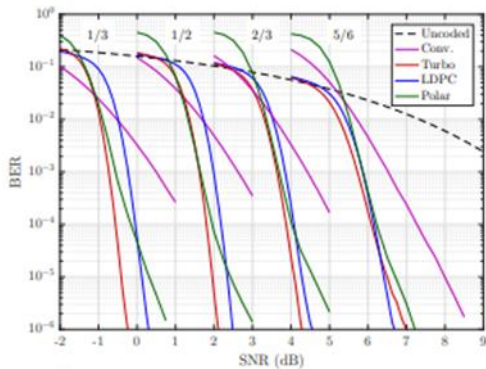
BER comparison for different code rates, $K = 2048$ (For LDPC, $K = 2052$ for $R = 1/2$, and $1/3$, and $K = 2040$ for $R = 5/6$.)



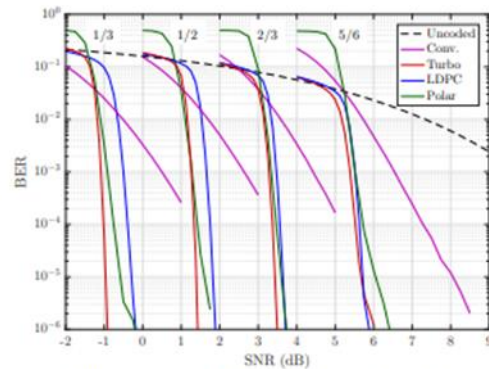
BER comparison for different code rates, $K = 512$ (For LDPC, $K = 516$ for $R = 1/2$, and $1/3$, and $K = 520$ for $R = 5/6$.)



BER comparison for different code rates, $K = 4096$ (For LDPC, $K = 4092$ for $R = 1/2$, and $1/3$, and $K = 4100$ for $R = 5/6$.)



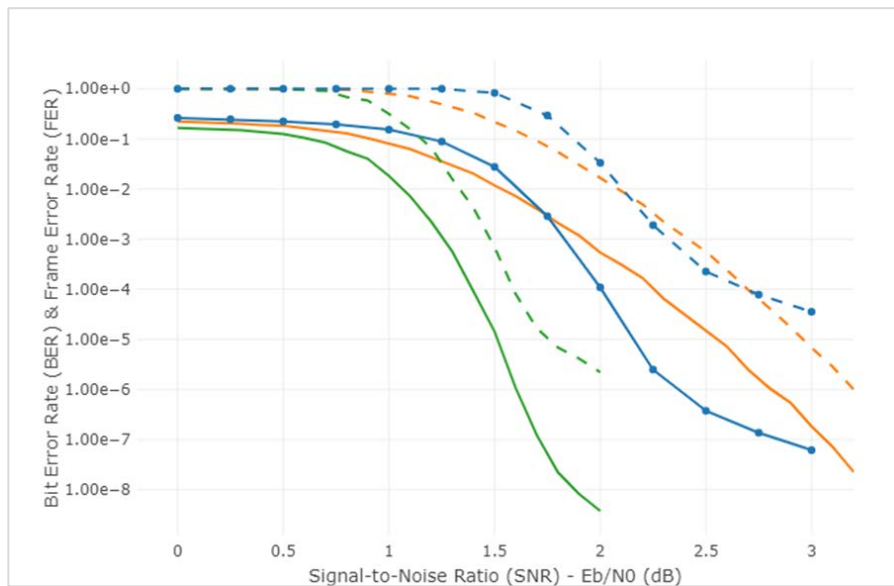
BER comparison for different code rates, $K = 1024$ (For LDPC, $K = 1020$ for $R = 1/2, 1/3$, and $5/6$.)



BER comparison for different code rates, $K = 8192$ (For LDPC, $K = 8196$ for $R = 1/2$, and $1/3$, and $K = 8200$ for $R = 5/6$.)

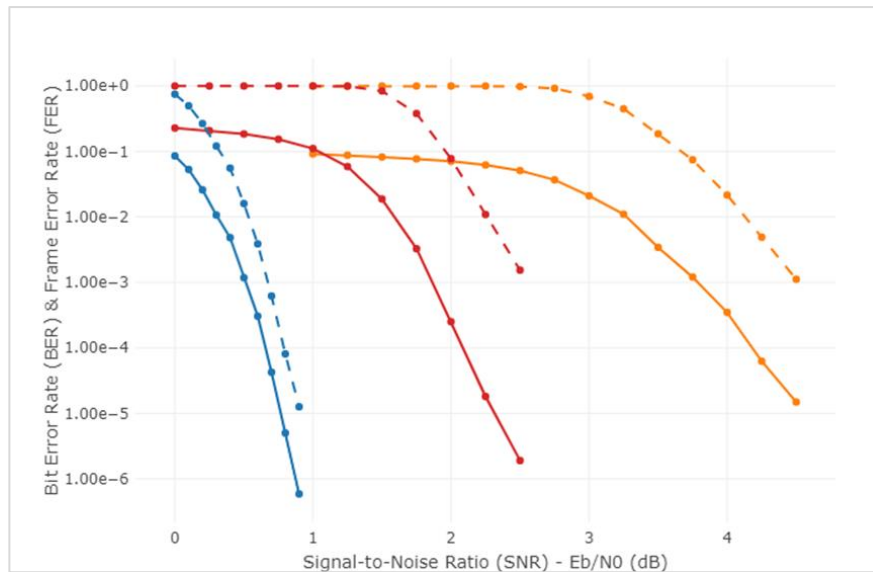
Σχήμα 43: Απόδοση των σχημάτων κωδικοποίησης. Πηγή: <https://aff3ct.github.io/>

Στα παραπάνω output, απεικονίζεται η απόδοση των σχημάτων κωδικοποίησης για μήκη block $K = 256, 512, 1024, 2048, 4096$ και 8192 και ρυθμό κώδικα $R = 1/3, 1/2, 2/3$ και $5/6$. Οι αλγόριθμοι αποκωδικοποίησης και ο αριθμός των επαναλήψεων ακολουθούν τα προηγούμενα output. Ο Turbo, ο LDPC και ο Polar αποδίδουν σχεδόν το ίδιο μεταξύ τους, για μεγάλα μήκη block. Τα αποτελέσματα του κώδικα Polar είναι πιο ενδιαφέρον διότι χρησιμοποιήθηκε μόνο ένα μέγεθος μέρος λίστας (8). Αυτό δείχνει ότι αν κατασκευαστεί κατάλληλα έχει εξαιρετικά αποτελέσματα. Όμως, μερικές καμπύλες του κώδικα Polar εμφανίζουν κορεσμό, για χαμηλές τιμές BER. Πιθανή βελτίωση μπορεί να επιτευχθεί αν διορθωθεί η κατασκευή του.



Σχήμα 44: $N = 4096, 4108, 4224$ και $R = 1/2$. Πηγή: <https://aff3ct.github.io/>

Παραπάνω απεικονίζεται η σύγκριση των τριών κωδικών Turbo, Polar και LDPC. Ο Turbo (4108,2048) παρουσιάζεται με την πράσινη καμπύλη με εύρος ζώνης 0 έως 2, ο LDPC (4224, 2112) με την μπλε καμπύλη με εύρος ζώνης 0 έως 3 και ο Polar (4096, 2048) με τη πορτοκαλί με εύρος ζώνης 0 έως 3,2. Η προσομοίωση πραγματοποιείται σε κανάλι AWGN και με ρυθμό $R = 1/2$. Στην συγκεκριμένη περίπτωση, καλύτερα αποδίδει ο κώδικας Turbo με $BER = 3,76e-9$ και $FER = 2,21e-6$. Ο LDPC τερματίζει με τις τιμές $BER = 6,15e-8$ και $FER = 3,55e-5$ και ο Polar με τις τιμές $BER = 2,24e-8$ και $FER = 9,99e-7$.



Σχήμα 45: Σύγκριση με διαφορετικό K,N,R,i. Πηγή: <https://aff3ct.github.io/>

Στην παραπάνω εικόνα απεικονίζεται ένας LDPC (4224,2112) GSM-SPA 10 turbo ite, ένας Turbo (5368, 1784) BCJR 10 ite. CCSDS και ένας Polar (2048, 1723) SC 8-bit. Σημαντικό ρόλο στην προσομοίωση είναι το frame size, όπου στην συγκεκριμένη περίπτωση πάλι αποδίδει καλύτερα. Άρα καλύτερος είναι ο Turbo με R=1/3, με i=10 επαναλήψεις και εύρος ζώνης από 0 έως 0,9.

```
# * Monitor -----
# ** Frame error count (e) = 100
# ** Compute mutual info = no
# * Terminal -----
# ** Enabled = yes
# ** Frequency (ms) = 500
#
# The simulation is running...
#
# Bit Error Rate (BER) and Frame Error Rate (FER) depending
# on the Signal Noise Ratio (SNR)
#
# -----||-----
# Es/N0 | Eb/N0 | FRA | BE | FE | BER | FER || CTHR | ITHR | LA
# (dB) | (dB) | | | | | | || (Mb/s) | (Mb/s) |
# -----||-----
#
# -4.78 | 0.00 | 184 | 28089 | 138 | 8.56e-02 | 7.50e-01 || 0.03 | 0.01 | 1.5
# -4.68 | 0.10 | 251 | 23802 | 125 | 5.32e-02 | 4.98e-01 || 0.04 | 0.01 | 1.5
# -4.58 | 0.20 | 420 | 19443 | 112 | 2.59e-02 | 2.67e-01 || 0.04 | 0.01 | 1.4
# -4.48 | 0.30 | 868 | 16582 | 105 | 1.07e-02 | 1.21e-01 || 0.04 | 0.01 | 1.3
# -4.38 | 0.40 | 1868 | 16132 | 104 | 4.84e-03 | 5.57e-02 || 0.04 | 0.01 | 1.3
# -4.28 | 0.50 | 6262 | 13216 | 100 | 1.18e-03 | 1.60e-02 || 0.04 | 0.01 | 1.2
# -4.18 | 0.60 | 25750 | 13996 | 100 | 3.05e-04 | 3.88e-03 || 0.04 | 0.01 | 1.2
# -4.08 | 0.70 | 161267 | 12187 | 100 | 4.24e-05 | 6.20e-04 || 0.04 | 0.01 | 1.2
# -3.98 | 0.80 | 1235902 | 11049 | 100 | 5.01e-06 | 8.09e-05 || 0.04 | 0.01 | 1.2
# -3.88 | 0.90 | 7959794 | 8342 | 100 | 5.87e-07 | 1.26e-05 || 0.04 | 0.01 | 1.1
#
# End of the simulation.
```

Παραπάνω απεικονίζεται το output του.

Αμέσως μετά ακολουθεί ο LDPC με $R=1/2$, $i=11$ επαναλήψεις και εύρος ζώνης 0 έως 2,5 dB. Ανάμεσα στον Turbo και LDPC υπάρχει κέρδος κωδικοποίησης 1,6dB.

```
# * Monitor -----
# ** Frame error count (e) = 100
# ** Compute mutual info = no
# * Terminal -----
# ** Enabled = yes
# ** Frequency (ms) = 500
#
# The simulation is running...
# -----||-----
# Bit Error Rate (BER) and Frame Error Rate (FER) depending || Decoder throughput ar
# on the Signal Noise Ratio (SNR) || latency (per thread)
# -----||-----
# -----||-----||-----||-----||-----||-----||-----||-----||-----||-----
# Es/N0 | Eb/N0 | FRA | BE | FE | BER | FER | CTHR | ITHR | LTHR
# (dB) | (dB) | | | | | | (Mb/s) | (Mb/s) | (Mb/s)
# -----||-----||-----||-----||-----||-----||-----||-----||-----||-----
# -3.01 | 0.00 | 147 | 70847 | 147 | 2.28e-01 | 1.00e+00 | 0.25 | 0.12 | 171
# -2.76 | 0.25 | 147 | 64538 | 147 | 2.08e-01 | 1.00e+00 | 0.26 | 0.13 | 161
# -2.51 | 0.50 | 147 | 57387 | 147 | 1.85e-01 | 1.00e+00 | 0.26 | 0.13 | 161
# -2.26 | 0.75 | 147 | 47851 | 147 | 1.54e-01 | 1.00e+00 | 0.26 | 0.13 | 161
# -2.01 | 1.00 | 147 | 34438 | 147 | 1.11e-01 | 1.00e+00 | 0.26 | 0.13 | 161
# -1.76 | 1.25 | 147 | 18252 | 146 | 5.88e-02 | 9.93e-01 | 0.26 | 0.13 | 166
# -1.51 | 1.50 | 170 | 6760 | 143 | 1.88e-02 | 8.41e-01 | 0.26 | 0.13 | 161
# -1.26 | 1.75 | 297 | 2052 | 113 | 3.27e-03 | 3.80e-01 | 0.26 | 0.13 | 166
# -1.01 | 2.00 | 1326 | 703 | 103 | 2.51e-04 | 7.77e-02 | 0.28 | 0.14 | 151
# -0.76 | 2.25 | 9268 | 354 | 101 | 1.81e-05 | 1.09e-02 | 0.29 | 0.15 | 145
# -0.51 | 2.50 | 64883 | 261 | 100 | 1.90e-06 | 1.54e-03 | 0.30 | 0.15 | 138
# End of the simulation.
```

Παραπάνω απεικονίζεται το output του LDPC.

Τέλος, είναι ο Polar με $R=1/2$, $i=15$ επαναλήψεις και εύρος ζώνης 0 έως 4,5. Ανάμεσα στον LDPC και Polar υπάρχει κέρδος κωδικοποίησης 2 dB και ανάμεσα στον Polar και Turbo 3,9dB.

```
# * Terminal -----
# ** Enabled = yes
# ** Frequency (ms) = 500
#
# The simulation is running...
# -----||-----
# Bit Error Rate (BER) and Frame Error Rate (FER) || Global throughput
# depending on the Signal Noise Ratio (SNR) || and elapsed time
# -----||-----
# -----||-----||-----||-----||-----||-----||-----||-----||-----||-----
# Es/N0 | Eb/N0 | FRA | BE | FE | BER | FER | SIM_THR | ET/R1
# (dB) | (dB) | | | | | | (Mb/s) | (hh:mm:ss)
# -----||-----||-----||-----||-----||-----||-----||-----||-----||-----
# 0.25 | 1.00 | 532 | 84173 | 532 | 9.18e-02 | 1.00e+00 | 6.28 | 00h00'00
# 0.50 | 1.25 | 536 | 80920 | 536 | 8.76e-02 | 1.00e+00 | 8.80 | 00h00'00
# 0.75 | 1.50 | 535 | 75875 | 535 | 8.23e-02 | 1.00e+00 | 8.72 | 00h00'00
# 1.00 | 1.75 | 536 | 71227 | 536 | 7.71e-02 | 1.00e+00 | 9.06 | 00h00'00
# 1.25 | 2.00 | 536 | 65426 | 536 | 7.08e-02 | 1.00e+00 | 10.02 | 00h00'00
# 1.50 | 2.25 | 535 | 57336 | 535 | 6.22e-02 | 1.00e+00 | 10.98 | 00h00'00
# 1.75 | 2.50 | 545 | 48081 | 536 | 5.12e-02 | 9.83e-01 | 11.21 | 00h00'00
# 2.00 | 2.75 | 580 | 36856 | 534 | 3.69e-02 | 9.21e-01 | 11.96 | 00h00'00
# 2.25 | 3.00 | 752 | 27246 | 522 | 2.10e-02 | 6.94e-01 | 15.61 | 00h00'00
# 2.50 | 3.25 | 1179 | 22251 | 529 | 1.10e-02 | 4.49e-01 | 24.37 | 00h00'00
# 2.75 | 3.50 | 2762 | 16320 | 514 | 3.43e-03 | 1.86e-01 | 55.40 | 00h00'00
# 3.00 | 3.75 | 6756 | 14102 | 504 | 1.21e-03 | 7.46e-02 | 126.52 | 00h00'00
# 3.25 | 4.00 | 23243 | 13960 | 501 | 3.49e-04 | 2.16e-02 | 335.54 | 00h00'00
# 3.50 | 4.25 | 101560 | 10960 | 500 | 6.26e-05 | 4.92e-03 | 732.08 | 00h00'00
# 3.75 | 4.50 | 447510 | 11476 | 500 | 1.49e-05 | 1.12e-03 | 989.43 | 00h00'00
# End of the simulation.
```

Και τέλος εδώ απεικονίζεται το output του Polar.

Κεφάλαιο 4: Συμπεράσματα

Βασικός στόχος της παρούσας εργασίας είναι η σύγκριση των τριών κωδικών κωδικοποίησης Turbo, Polar και LDPC. Μέσα από μία αναλυτική παρουσίαση των τριών κωδικών, της προβολής του τρόπου λειτουργίας και των μεθόδων κωδικοποίησης και αποκωδικοποίησης, των προσομοιώσεων και των διαφορών διακρίνονται οι ομοιότητες και οι αντιθέσεις τους.

Ο κώδικας Turbo ήταν ο πρώτος πρακτικός κώδικας που προσέγγισε το μέγιστο κανάλι χωρητικότητας ή όριο Shannon και αυτό ήταν κάτι πολύ επαναστατικό για την εποχή του. Στην πορεία ανακαλύφθηκαν οι κώδικες LDPC και Polar, οι οποίοι με χαμηλότερη πολυπλοκότητα κατάφεραν να έχουν εφαρμογή στα επερχόμενα δίκτυα 5^{ης} γενιάς (5G) και να αποτελούν το μέλλον της εξέλιξης της θεωρίας των κωδικών.

Η εργασία λοιπόν, αποτελεί μία εισαγωγή σε αυτό το νέο πεδίο έρευνας της θεωρίας της κωδικοποίησης. Παρόλα αυτά, έχει την δυνατότητα να επεκταθεί περαιτέρω σχετικά με τα ανοιχτά ζητήματα και τις νέες απαιτήσεις που επιβάλλουν οι εξελίξεις της εποχής.

Βιβλιογραφία

Ξενόγλωσση

Bernard M. J. Leiner, “LDPC Codes – a brief Tutorial”, April 8 2005

Branka Vucetic, Jinhong Yuan, “Turbo Codes Principles and Applications”, Kluwer Academic Publishers, 2010.

David J. C. MacKay, Radford M. Neal, “Near Shannon Limit Performance of Low Density Parity Check Codes”,
July 12 1996

Διαδικτυακή

<http://www.alexiosbalatsoukas.com/papers/17WCNCPolarCompSlides.pdf>. Πρόσβαση στις
10/01/2024.

<https://www.icpsr.umich.edu/icpsrweb/content/shared/ICPSR/faqs/what-is-a-codebook.html>.
Πρόσβαση στις 10/01/2024.

https://eclass.uoa.gr/modules/document/file.php/D63/1_Introduction-TM_Oct-2015%20%5BCompatibility%20Mode%5D.pdf. Πρόσβαση στις 10/01/2024.

<https://www.computerworld.com/article/3228804/how-polar-codes-work.html>. Πρόσβαση στις
10/01/2024.

<https://www.retailnews.asia/huawei-achieves-27gbps-5g-speeds-polar-code/>. Πρόσβαση στις
12/01/2024.

<https://www.telecomreview.com/index.php/articles/telecom-vendors/2062huawei-partners-with-telus-on-5g-wireless-to-the-home-trial>. Πρόσβαση στις 12/01/2024.

<https://www.dsprelated.com/showarticle/1201.php>. Πρόσβαση στις 12/01/2024.

https://eprints.soton.ac.uk/427712/1/Survey_of_Turbo_LDPC_and_Polar_Decoder_ASIC_Implementations.pdf. Πρόσβαση στις 13/01/2024.

https://www.researchgate.net/figure/Performance-comparison-between-Turbo-code-O-and-LDPC-for-rate-1-2-and-coded_fig2_234051399. Πρόσβαση στις 15/01/2024.

<https://aff3ct.readthedocs.io/en/latest/others/readme/readme.html>. Πρόσβαση στις 16/01/2024.

file:///C:/Users/user/Downloads/Polarizationadjusted_Convolutional_PAC_Codes_Sequ.pdf.
Πρόσβαση στις 17/01/2024.

https://www.researchgate.net/figure/AFF3CT-simulation-of-various-code_families_fig2_337176980.
Πρόσβαση στις 17/01/2024.

<https://aff3ct.github.io/>. Πρόσβαση στις 19/01/2024.

<https://imtech.imt.fr/en/2016/09/16/what-are-turbo-codes/>. Πρόσβαση στις 19/01/2024.

http://www.scholarpedia.org/article/Turbo_code. Πρόσβαση στις 20/01/2024.

<https://slideplayer.com/slide/10909204/>. Πρόσβαση στις 20/01/2024.

https://www.researchgate.net/figure/Performance-comparison-between-Turbo-code-O-and-LDPC-for-rate-7-8_fig1_234051399. Πρόσβαση στις 21/01/2024.

<https://docplayer.gr/71144328-Tmima-mihanikon-i-y-kai-pliroforikis.html>. Πρόσβαση στις 22/01/2024.