ANDREW AND ERNA VITERBI FACULTY OF ELECTRICAL ENGINEERING

הטכניון מכון טכנולוגי לישראל TECHNION Israel Institute of Technology

COMMUNICATION LABORATORY
המעבדה לתקשורת

חומר רקע - קודים לתיקון שגיאות

מעבדה 2,3 בהנדסת חשמל

הפקולטה להנדסת חשמל ומחשבים
ע"ש ויטרבי

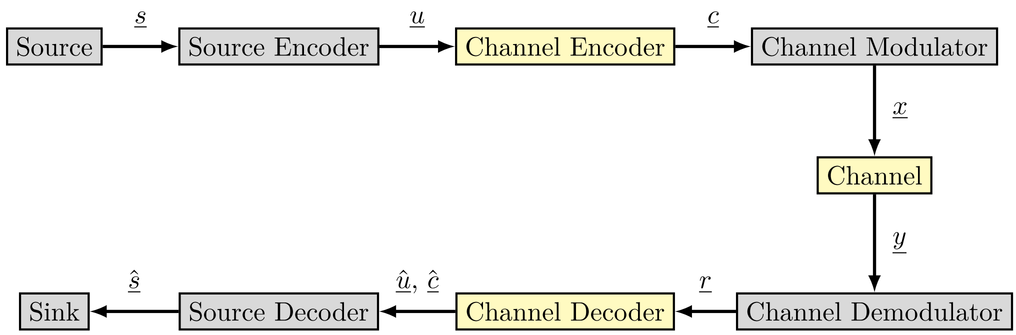
הטכניון – מכון טכנולוגי לישראל

נכתב על-ידי: אשד רם, אוגוסט 2019.

עדכון אחרון: מיכאל דיקשטיין, 18 במרץ 2022.

תוכן עניינים

רשימת איורים



איור 1: מבנה כללי של מערכת תקשורת. המסמך עוסק בחלקים הצהובים בלבד.

1 מבוא

כיום, כמעט כל מכשיר אלקטרוני מתקשר עם סביבתו: החל ממחשבים אשר מתקשרים עם מחשבים אחרים דרך האינטרנט באמצעות תקשורת אלחוטית (WiFi) או חוטית (Ethernet), דרך טלפונים סלולרים (BlueTooth, Cellular), לוויינים וגשושיות בחלל (Deep Space Communication), ועד למכוניות אוטונומיות והאינטרנט של הדברים(IoT - Internet of Things). בכל אחת מהדוגמאות לעיל מידע עובר בין משתמשים בתווך פיזיקלי מסוים – אוויר, כבלים אופטיים וכדומה – אשר עלול לשבש את האות העובר דרכו; אנחנו קוראים לתווך הזה **ערוץ רועש**. כמובן שנרצה שהודעה שנשלחה ממשתמש א' תגיע למשתמש ב' כפי שנשלחה ולא עם שגיאות. כאן נכנסים לתמונה קודים לתיקון שגיאות (ECC - Error Correcting Codes). למעשה, קודים לתיקון שגיאות הם ההבדל בין מערכת תקינה למערכת לא פונקציונלית. קודים לתיקון שגיאות נמצאים בכל תקני התקשורת הקיימים וגם בהתקני זיכרון כמו DVD, SSD, Magnetic Recording, DNA Storage כדי להגן על המידע שמאוכסן בהם מפני רעש.

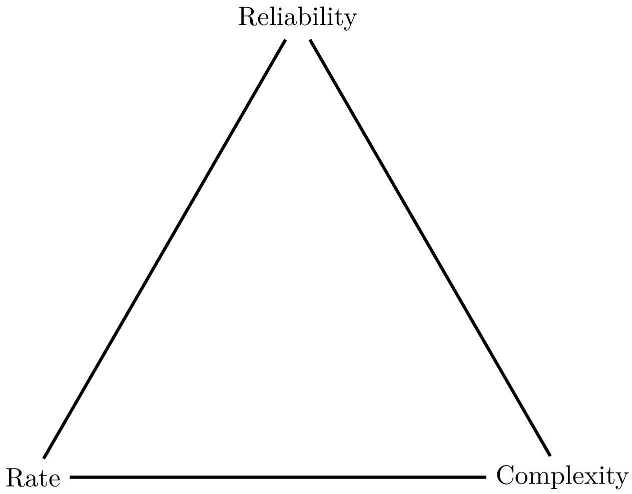
נניח תקשורת בין 2 משתמשי קצה: משתמש א' ומשתמש ב'. תפקידה של מערכת התקשורת הינה לבצע פעולות על המידע שנשלח בין המשתמשים כדי להעביר אותו בצורה אמינה, כלומר ללא שגיאות, בצורה יעילה וחסכונית ככל שניתן. מבנה כללי של מערכת תקשורת כזו מודגמת באיור 1. בניסוי נעסוק בחלקים שצבועים בצהוב: **מקודד ערוץ, ערוץ, ומפענח הערוץ**. מקודד הערוץ מקבל בכניסה שלו **מילת אינפורמציה** $\underline{u}$, מגן עליה בעזרת הוספת יתירות ובמוצא נותן כפלט **מילת קוד** $\underline{c}$. הערוץ מקבל אות פיזיקלי (מילת קוד שעברה מודולציה) בכניסה $\underline{x}$, משבש אותו בצורה אקראית ומוציא את האות $\underline{y}$ אשר עובר דה־מודולציה. לתוך מפענח הערוץ נכנס **האות הנקלט** $\underline{r}$ והוא מחשב את **מילת הקוד המשוערכת** $\hat{\underline{c}}$ ואת מילת האינפורמציה המתאימה $\hat{\underline{u}}$.

איור 1 מציג גם נושאים שלא נעסוק בהם במסמך זה, כמו קידוד מקור (Source Coding) שמטרתו לדחוס מידע, וביצוע מודולציה ודה־מודולציה לערוצים. חשוב לציין שישנם מקרים שבהם מאחדים בין חלקים מסוימים באיור 1.

אנחנו נסמן את **ספר הקוד** (אוסף כל **מילות הקוד** האפשריות) בסימון $\mathcal{C}$, את **מספר מילות הקוד** ב–$M = |\mathcal{C}|$, ואת **אורך הקוד** (אורך הווקטור $\underline{c}$) ב–n. בניסוי זה נעסוק לצורך פשטות בקודים בינאריים בלבד, כלומר $\mathcal{C} \subseteq \{0,1\}^n$, אך חשוב לציין שקודים לא בינאריים נמצאים בהרבה מערכות תקשורת מודרניות. **קצב הקוד** מוגדר להיות

$$R \triangleq \frac{\log_2 M}{n},$$

ונמדד ביחידות של ביטים לשימושי ערוץ.



איור 2: דרישות בתכנון מערכת תקשורת.

האתגר של מתכנן סכימת הקידוד הוא להציע את השלישיה **קוד/מקודד/מפענח** אשר יעמדו בדרישות של אמינות גבוהה (High Reliabilty), קצב גבוה (High Rate) וסיבוכיות קידוד ופיענוח נמוכים (Low Complexity) (ראה איור 2).

2 ערוצים רועשים

הדבר הראשון שיש להבין בעת תכנון מערכת לתיקון שגיאות הינו מודל הרעש/השגיאה שמולו אנו מתגוננים. המודל בו אנו נעסוק הינו מודל הסתברותי, כלומר נניח שהרעש במערכת התקשורת הינו אקראי ולא זדוני (Adversarial).

הגדרה 1 ערוץ רועש מוגדר בעזרת השלישיה $\mathcal{X}, \mathcal{Y}, \Pr$ כאשר:

1. $\mathcal{X}$ הינו א"ב הכניסה לערוץ.
2. $\mathcal{Y}$ הינו א"ב המוצא מהערוץ.
3. $\Pr(y|x)$ הינה הסתברות המעבר של הערוץ, $x \in \mathcal{X},\ y \in \mathcal{Y}$.

$$x \in \mathcal{X} \longrightarrow \boxed{\text{Noisy Channel} - \Pr(y|x)} \longrightarrow y \in \mathcal{Y}$$

איור 3: הערוץ הרועש

דוגמה 1 בערוץ הבינארי סימטרי עם הפרמטר $p \in [0,1]$, $BSC(p)$ – Binary Symmetric Channel מתקיים $\mathcal{X} = \mathcal{Y} = \{0,1\}$ והסתברות המעבר היא

$$\Pr(y|x) = \begin{cases} p & y \neq x \\ 1-p & y = x \end{cases} = p^{x\oplus y}(1-p)^{1-x\oplus y},$$

כאשר $\oplus$ זה סימון ל־XOR.

דוגמה 2 בערוץ המחיקה הבינארי עם הפרמטר $p \in [0,1]$, $BEC(p)$ – Binary Erasure Channel מתקיים $\mathcal{X} = \{0,1\},\ \mathcal{Y} = \{0,?,1\}$, והסתברות המעבר היא

$$\Pr(y|x) = \begin{cases} p & y = ? \\ 1-p & y = x \end{cases}.$$

דוגמה 3 בערוץ הגאוסי האדיטיבי [1] עם הפרמטר $\sigma > 0$, $AWGN(\sigma)$ – Additive White Gaussian Noise [2] מתקיים $\mathcal{X} = \mathcal{Y} = \mathbb{R}$ והסתברות המעבר היא

$$\Pr(y|x) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(x-y)^2}{2\sigma^2}}.$$

מכיוון שמידע המועבר הינו דיגיטלי, אז צריך לתרגם אותו לאות רציף בכניסה לערוץ [3]. פעולה זו נקראת מודולציה, ואנחנו נתמקד בסוג ספציפי של מודולציה שנקראת BPSK – Binary Phase Shift Keying, שבה מתבצע המיפוי הבא

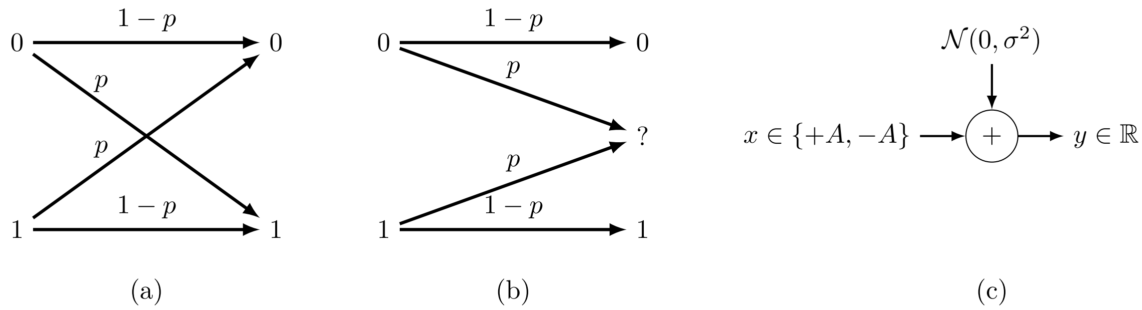
$$'0' \to +A, '1' \to -A,$$

עבור A חיובי כלשהו. במילים אחרות, נגביל את א"ב הכניסה להיות $\mathcal{X} = \{+A, -A\}$.

[1] למעשה, הנחנו כאן מודלוצית (איפנון) BPSK. קיימות גם מודולציות אחרות שלא נעמיק בהם כאן.

[2] שונות המשתנה הגאוסי מתקשרת לצפיפות ההספק הספקטרלית (N_0) של הרעש הלבן הגאוסי דרך $\sigma^2 = \frac{N_0}{2}$

[3] למען הדיוק, המידע מקודד לאותות רציפים בזמן. פעולה זו מתבצעת במרחב אותות אורתוגנלים, והאות שמייצג את המידע הדיגיטלי מורכב מצירוף ליניארי של אותות הבסיס, כאשר המקדמים הם מעל קבוצה סופית (הקונסטלציה) והם מייצגים את המידע



איור 4: שלושת הערוצים מהדוגמאות: ערוץ בינארי סימטרי (a), ערוץ מחיקה בינארי (b), ערוץ גאוסי אדיטיבי עם מיפוי BPSK ‏(c).

במקרה של הערוץ הגאוסי, המשתנה A הינו עוצמת שידור. עבור שונות רעש נתונה σ^2 הגדלת עוצמת השידור תביא להסתברות שגיאה נמוכה יותר (כי השידור גובר על הרעש) אבל מצד שני מגדילה את האנרגיה המושקעת בשידור. יש כאן למעשה trade-off בין **אנרגיה** מושקעת להסתברות שגיאה. האנרגיה של אות השידור $\underline{x} = (x_1, x_2, \ldots, x_n) \in \mathbb{R}$ מוגדרת בתור

$$E(\underline{x}) = \sum_{i=1}^{n} x_i^2$$

ובמקרה של מיפוי BPSK נקבל $E(\underline{x}) = A^2 \cdot n$.

הגדרה 2 ערוץ ייקרא **חסר־זכרון** אם לכל n מתקיים

$$\Pr(\underline{y}|\underline{x}) = \prod_{i=1}^{n} \Pr(y_i|x_i),\ \forall \underline{y} \in \mathcal{Y}^n,\ \forall \underline{x} \in \mathcal{X}^n.$$

במילים, בערוץ חסר־זכרון הכניסה בזמן $i \in \{1, 2, \ldots, n\}$ משפיעה על המוצא בזמן i בלבד.

3 שדות סופיים

הרבה קודים לתיקון שגיאות מבוססים על מושג מתמטי שנקרא שדה סופי.

הגדרה 3 תהי F קבוצה ויהיו $+$ ו-$\cdot$ שתי פעולות ("חיבור" ו"כפל") שמוגדרות על זוג איברים ב־F. השלישיה $(F,+,\cdot)$ תיקרא שדה אם:

1. חיבור:
   (א) סגירות: $\forall a,b\in F,\ a+b\in F$.
   (ב) קומוטטיביות: $\forall a,b\in F,\ a+b=b+a$.
   (ג) אסוציאטיביו,: $\forall a,b,c\in F,\ a+(b+c)=(a+b)+c$.
   (ד) קיום אדיש חיבורי: $0\in F,\ \forall a\in F,\ a+0=a=0+a$.
   (ה) קיום הופכי: $\forall a\in F,\ \exists b\in F,\ a+b=0=b+a$.
2. כפל:
   (א) סגירות: $\forall a,b\in F\setminus\{0\},\ a\cdot b\in F\setminus\{0\}$.
   (ב) קומוטטיביות: $\forall a,b\in F\setminus\{0\},\ a\cdot b=b\cdot a$.
   (ג) אסוציאטיביות: $\forall a,b,c\in F\setminus\{0\},\ a\cdot(b\cdot c)=(a\cdot b)\cdot c$.
   (ד) קיום אדיש כפלי: $\exists 1\in F\setminus\{0\},\ \forall a\in F\setminus\{0\},\ a\cdot 1=a=1\cdot a$.
   (ה) קיום הופכי: $\forall a\in F\setminus\{0\},\ \exists b\in F\setminus\{0\},\ a\cdot b=1=b\cdot a$.
3. דיסטריביוטיביות: $\forall a,b,c\in F\setminus,\ a\cdot(b+c)=(a\cdot b)+(a\cdot c)$.

דוגמה 4 דוגמא לשדה היא שדה הממשיים עם $F=\mathbb{R}$, ופעולות כפל וחיבור הידועים לנו. עוד דוגמא היא שדה המספרים הרציונלים $F=\mathbb{Q}$, ופעולות כפל וחיבור הידועים לנו.

דוגמה 5 קבוצת השלמים $F=\mathbb{Z}=\{\ldots,-2,-1,0,1,2,\ldots\}$ עם כפל וחיבור רגילים איננה שדה מכיוון שלא קיים הופכי כפלי, למשל ל־2.

הגדרה 4 **שדה סופי** הינו שדה בעל מספר סופי של איברים $(|F|<\infty)$.

דוגמה 6 דוגמא לשדה סופי היא השדה הבינארי עם $F=\{0,1\}$ וחיבור וכפל מודולו 2.

דוגמה 7 הקבוצה $F=\{0,1,2,3\}$ עם פעולת החיבור והכפל המוגדרות כך:

$+$	0	1	2	3
0	0	1	2	3
1	1	0	3	2
2	2	3	0	1
3	3	2	1	0

$\cdot$	0	1	2	3
0	0	0	0	0
1	0	1	2	3
2	0	2	3	1
3	0	3	1	2

בדוגמא הזאת, האדיש החיבורי הוא 0, האדיש הכפלי הוא 1, ההופכי החיבורי של 2 הוא 2 וההופכי הכפלי של 2 הוא גם 3.

משפט 1

1. לכל שדה סופי יש q^m איברים כאשר q מספר ראשוני, ו־m מספר שלם.
2. לכל q ראשוני ו־m שלם ניתן למצוא שדה סופי עם q^m איברים.

דוגמה 8 לא ניתן ליצור שדה סופי עם 6 איברים, אבל קיים שדה סופי עם 256 איברים.

נהוג לסמן שדה סופי בעזרת הגודל שלו בצורה הבאה: $F=GF(q^m)$ (Galois Field). למשל, עבור השדה הבינארי נסמן $F=GF(2)$.

4 קודים

בפרק זה נגדיר פורמלית מה זה קוד, ונדון במאפיינים הקשורים לבעיית תיקון השגיאות.

הגדרה 5 יהי F שדה סופי, ויהי n שלם חיובי. קוד בלוק באורך n מעל F הינו אוסף מילים $\mathcal{C} \subset F^n$. גודל הקוד מוגדר כ־ $M = |\mathcal{C}|$, וקצב הקוד הוא $R = \frac{\log_{|F|} M}{n}$.

דוגמה 9 קוד חזרות באורך n מעל שדה כלשהו F מוגדר להיות

$$\mathcal{C}_{rep} = \{(x_1, x_2, \ldots, x_n) \in F^n \colon x_1 = x_2 = \cdots = x_n\} \,,$$

יש בו $M = |F|$ מילים, וקצב הקוד הוא $R = \frac{1}{n}$. לדוגמא, קוד חזרות בינארי באורך 7 מורכב מ־2 מילים: $(0,0,0,0,0,0,0)$, ו־ $(1,1,1,1,1,1,1)$.

דוגמה 10 קוד זוגיות באורך n מעל שדה כלשהו F מוגדר להיות

$$\mathcal{C}_{par} = \{(x_1, x_2, \ldots, x_n) \in F^n \colon x_1 + x_2 + \ldots x_n = 0\}$$

יש בו $M = |F|^{n-1}$ מילים, וקצב הקוד הוא $R = \frac{n-1}{n}$. לדוגמא, קוד זוגיות בינארי באורך 3 מורכב מ־4 מילים שיש בהם מספר זוגי של אחדים: $(1,1,0)$, $(1,0,1)$, $(0,0,0)$, ו־ $(1,1,0)$.

הגדרה 6 יהי F שדה סופי, ויהי n שלם חיובי. לכל שני וקטורים $\underline{x}, \underline{y} \in F^n$ נגדיר את מרחק Hamming ביניהם בתור מספר הקואורדינטות שבהם הם שונים:

$$d_H(\underline{x}, \underline{y}) = |\{1 \leq i \leq n \colon x_i \neq y_i\}| \,,$$

ולכל וקטור $\underline{x} = (x_1, x_2, \ldots, x_n) \in F^n$ נגדיר את משקל Hamming בתור

$$w_H(\underline{x}) = d_H(\underline{x}, \underline{0}) \,.$$

מרחק Hamming מקיים את שלושת התכונות שמטריקה אמורה לקיים:

1. אי־שליליות: $\forall \underline{x}, \underline{y} \in F^n,\ d_H(\underline{x}, \underline{y}) \geq 0$ $\left(d_H(\underline{x}, \underline{y}) = 0 \iff \underline{x} = \underline{y}\right)$.
2. סימטריות: $\forall \underline{x}, \underline{y} \in F^n,\ d_H(\underline{x}, \underline{y}) = d_H(\underline{y}, \underline{x})$.
3. אי־שיוויון המשולש: $\forall \underline{x}, \underline{y}, \underline{z} \in F^n,\ d_H(\underline{x}, \underline{z}) \leq d_H(\underline{x}, \underline{y}) + d_H(\underline{y}, \underline{z})$.

הגדרה 7 לכל קוד $\mathcal{C}$ נגדיר את **מרחק הקוד** בתור

$$\mathrm{d}(\mathcal{C}) = \min_{\substack{\underline{c}_1, \underline{c}_2 \in \mathcal{C} \\ \underline{c}_1 \neq \underline{c}_2}} d_H(\underline{c}_1, \underline{c}_2) \,.$$

דוגמה 11 מרחקי קודי חזרות וזוגיות באורך n הינם $\mathrm{d}(\mathcal{C}_{rep}) = n$, ו־ $\mathrm{d}(\mathcal{C}_{par}) = 2$, בהתאמה.

קודים לינארים

הגדרה 8 יהי F שדה סופי, ויהי n שלם חיובי. קוד $\mathcal{C}$ מעל F באורך n ייקרא לינארי אם הוא סגור לחיבור וכפל בסקלר ב־F: $\alpha\underline{c}_1 + \beta\underline{c}_2 \in \mathcal{C}, \quad \forall, \underline{c}_1, \underline{c}_2 \in \mathcal{C}, \quad \forall\alpha, \beta \in F$.

במקרה זה, $\mathcal{C}$ הינו תת־מרחב לינארי של F^n, ונסמן את מימד תת־המרחב (הקוד) ב–k. מאלגברה לינארית אנחנו יודעים שכל ווקטור בתת־המרחב הלינארי הזה (כלומר כל מילת קוד) ניתן לכתיבה כצירוף לינארי של איברי בסיס המרחב. מכיוון שיש בדיוק $M = |F|^k$ צירופים לינאריים, אז נסיק שיש $M = |F|^k$ מילות קוד, וקצב הקוד הוא $R = \frac{k}{n}$.

משפט 2 מרחק קוד ליניארי נתון ע"י: $\mathrm{d}(\mathcal{C}) = \min_{\substack{\underline{c}\in\mathcal{C} \\ \underline{c}\neq\underline{0}}} \mathrm{w}_H(\underline{c})$.

נהוג לסמן את הפרמטרים של קוד לינארי בתוך סוגריים מרובעים: $[n, k, d]$.

משפט 3 (חסם סינגלטון) לכל קוד ליניארי $[n, k, d]$ מתקיים $d \leq n - k + 1$.

חסם סינגלטון מבטא את ה־ trade-off המובנה שיש בקודים לתיקון שגיאות. אם רוצים להגדיל את מרחק הקוד (על־מנת לשפר את יכולות התיקון של הקוד – ראו בהמשך פרק 6) צריך לשלם בקוד יותר ארוך (להגדיל את n) או להפחית את כמות האינפורמציה שניתן להעביר דרך הקוד (להקטין את k). קודים שמשיגים את חסם סינגלטון בשיוויון נקראים קודי MDS – Maximum Distance Seprable. דוגמא אחת לקודי MDS הם קודי RS – Reed-Solomon שקרויים על שם ממציאיהם, והם אחת ממשפחות הקודים הכי נפוצות כיום. אחד המאפיינים של קודי Reed-Solomon הוא שהם אינם בינאריים, כלומר מילת קוד מורכבת מסימבולים שהם מעל שדה סופי $F = GF(q)$ כאשר $q > 2$. למעשה, הקוד מחייב ש־ $q > n$, ולכן, כדי לקבל קודים ארוכים מאוד (ובכך להוריד את הסתברות השגיאה) יש צורך בשדות גדולים מאוד. דבר זה עלול להוות חסרון מבחינת סיבוכיות חישובית. בפועל, רוב הפעמים קודי Reed-Solomon מוגדרים מעל שדה סופי בגודל שהינו חזקה שלמה של 2 – $F = GF(2^m)$. זה מתאים מבחינה פרקטית למערכות מחשב מכיוון שאז ניתן להתייחס לרצף של m ביטים בתור איבר בשדה.

עוד משפחה מפורסמת של קודי לינארים הינם קודי BCH – Bose–Chaudhuri–Hocquenghem שגם כן קרויים על שם ממציאיהם. בניגוד לקודי RS, קודי BCH אינם קודי MDS. לעומת זאת, הם מוגדרים מעל שדות קטנים יותר, ולכן הם אטרקטיביים לאפליקציות מסוימות. ספציפית, קודי ה־BCH הנפוצים הינם בינארים. ישנו מספר עצום של משפחות של קודים לינארים, המפורסמים שבהם הם קודי RS Codes, BCH Codes, Low-Density Parity-Check (LDPC) Codes, Polar Codes, Convolutional Codes.

הגדרה 9 יהי $\mathcal{C}$ קוד לינארי באורך n מעל שדה F. **הקוד הדואלי** של $\mathcal{C}$ מסומן על־ידי $\mathcal{C}^{\perp}$ ומוגדר להיות

$$\mathcal{C}^{\perp} = \left\{ \underline{x} = (x_1, x_2, \ldots, x_n) \in F^n \colon \underline{x} \cdot \underline{c} = \sum_{i=1}^{n} x_i c_i = 0, ; \forall \underline{c} = (c_1, c_2, \ldots c_n) \in \mathcal{C} \right\}.$$

במילים, הקוד הדואלי ל־ $\mathcal{C}$ הינו אוסף כל המילים אשר "אורתוגונליות" לכל מילות הקוד ב־$\mathcal{C}$. מתקיים שגם $\mathcal{C}^{\perp}$ הוא קוד לינארי, ואם מימד הקוד $\mathcal{C}$ הינו k, אז מימד הקוד $\mathcal{C}^{\perp}$ הוא $n - k$.

דוגמה 12 הקוד הדואלי לקוד חזרות באורך n הינו קוד זוגיות באורך n, ולהיפך (בדקו!).

5 מקודדים

באופן פורמלי, מקודד הינו מיפוי חד־חד־ערכי ועל $\mathcal{E}\colon \mathcal{U} \to \mathcal{C}$, כאשר $\mathcal{U}$ זה אוסף כל מילות האינפורמציה בכניסה למקודד ו–$\mathcal{C}$ זה ספר הקוד. המיפוי הינו חד־חד־ערכי מכיוון שלא נרצה למפות 2 הודעות שונות לאותה מילת קוד (אחרת, איך נבדיל ביניהן), והוא על מכיוון שהם לא אז יש מילת קוד מיותרת שלא משתמשים בה – נקטין את ספר הקוד.

המקודד מוסיף למילת האינפרומציה $\underline{u} \in \mathcal{U}$ סימבולים של יתירות. אם אורך מילת האינפורמציה בכניסה למקודד הינו k, אז המקודד מוסיף $r = n - k$ סימבולי יתירות ויוצר את מילת הקוד $\underline{c} \in \mathcal{C}$. במקרה זה קצב הקוד הינו $R = \frac{k}{n}$.

כל מימוש של מקודד יכלול מימוש של המיפוי $\mathcal{E}$. מימוש נאיבי של מקודד ישמור טבלה שבה יש M שורות, ובכל שורה יש מילת אינפורמציה $\underline{u} \in \mathcal{U}$ ומילת הקוד המתאימה $\mathcal{E}(\underline{u})$. במקרה שהקוד $\mathcal{C}$ הינו קוד ליניארי מעל שדה F, אז כל מילת קוד הינה צירוף לינארי של ווקטורי הבסיס. מכיוון שיש k ווקטורי בסיס $\{\underline{e}_1, \underline{e}_2, \ldots, \underline{e}_k\}$, אז דרושים k מקדמים מעל F לצירוף הלינארי הנ"ל. נסמן ב–

$$G = \begin{pmatrix} e_{1,1} & e_{1,2} & \cdots & e_{1,n} \\ e_{2,1} & e_{2,2} & \cdots & e_{2,n} \\ \vdots & \vdots & \ddots & \vdots \\ e_{k,1} & e_{k,2} & \cdots & e_{k,n} \end{pmatrix}_{k \times n}$$

מטריצה ששורותיה מהוות בסיס לקוד $\mathcal{C}$; מטריצה זו נקראת **מטריצה יוצרת** (Genarator Matrix) של הקוד $\mathcal{C}$. אז כל מילת קוד $\underline{c} \in \mathcal{C}$, ניתנת לכתיבה כ–$\underline{c} = \sum_{i=1}^{k} u_i \underline{e}_i$ עבור $\{u_i\}_{i=1}^{k}$ סקלארים כלשהם. בכתיב מטריצי נקבל ביטוי למיפוי של המקודד למקרה הלינארי

$$\underline{c} = \mathcal{E}(\underline{u}) = \underline{u}G.$$

אם המטריצה היוצרת היא מהצורה $G = (I_k \mid A)$, כאשר I_k היא מטריצת היחידה בגודל $k \times k$ ו־ $\mid$ מסמן שרשור אופקי, אז נאמר ש-G הינה **סיסטמטית**, ונקבל שמילת האינפורמציה $\underline{u}$ נמצאת ב–k האינדקסים הראשונים של מילת הקוד $\underline{c}$

$$\underline{c} = \underline{u}\,(I_k \mid A) = (\underline{u} \mid \underline{u}A)\,.$$

דוגמה 13 נניח מטריצה יוצרת לקוד בינארי

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

אורך הקוד הינו $n = 7$, השורות הינם בלתי תלויות, אז מימד הקוד $k = 3$ ויש בקוד $M = |F|^k = 2^3 = 8$ מילים, לפי המיפוי הסיסטמתי הבא:

in ($\underline{u}$)	out ($\underline{c} = \underline{u}G$)
000	0000000
001	0011101
010	0101011
011	0110110
100	1000111
101	1011010
110	1101100
111	1110001

מרחק הקוד לפי משפט 2 הינו $d = 4$. לסיכום, הקוד שמוגדר על־ידי G הינו קוד $[7, 3, 4]$.

הערה 1 כאשר התקשורת מבוצעת על גבי הערוץ הגואסי עם מיפוי BPSK $'0' \to +A,\ '1' \to -A$ (ראו פרק 2), אז קצב הקוד מקבל משנה תוקף מבחינת אנרגיה שמושקעת בשידור. נניח שמוקצה לכם תקציב של E_b יחידות אנרגיה לכל ביט **אינפורמציה** שאתם משדרים. כלומר, למילת קוד בינארית באורך n, שבה יש k ביטי אינפורמציה, מוקצה $k \cdot E_b$ יחידות אנרגיה. בפועל, האנרגיה הכוללת של האות המשודר (מילת קוד) הינה (ראו פרק 2) $E = A^2 n$. מהשוואת הביטוים נקבל ש

$$A = \sqrt{\frac{k}{n} E_b} = \sqrt{R \cdot E_b}\,.$$

כלומר, קוד עם הרבה יתירות (קצב נמוך) יגרור עוצמת שידור נמוכה יותר, וזה עלול דווקא להעלות את הסתברות השגיאה!

אם נסמן ב־H את **המטריצה היוצרת של הקוד הדואלי** $\mathcal{C}^{\perp}$, אז נקבל ש־

$$\mathcal{C} = \left\{\underline{c}\colon H\underline{c}^T = \underline{0}\right\}.$$

מטריצה זו נקראת **מטריצת בדיקת זוגיות** (Parity-Check Matrix) של הקוד $\mathcal{C}$, שכן בעזרתה ניתן לבדוק האם וקטור מסוים הוא מילת קוד או לא (הכפלה מימין במטריצה נותנת 0 אם ורק אם הווקטור הוא מילת קוד). למעשה, הקוד $\mathcal{C}$ הינו הגרעין (kernel) של המטריצה H.

הערה 2 שימו לב שפעולות הכפל המטריציוניות לעיל – עם G או עם H – מתבצעות מעל השדה הסופי F. כלומר, על פעולות החיבור והכפל לציית להגדרת השדה. למשל, אם השדה הינו בינארי, אז כל פעולות החיבור הן למעשה XOR.

6 מפענחים

בהרבה מובנים, הפיענוח הינו האתגר הגדול ביותר של מתכנן סכימת הקידוד. מצד אחד, ישנה הדרישה להסתברות שגיאת פיענוח נמוכה מאוד, ומצד שני מהירות הפיענוח צריכה להיות גבוהה (כלומר סיבוכיות נמוכה).

פורמלית, המפענח הינו מיפוי בין מוצא הערוץ לספר הקוד $\mathcal{D}\colon \mathcal{Y}^n \to \mathcal{C}$. כלומר, עבור $\underline{r} \in \mathcal{Y}^n$ וקטור מוצא מהערוץ, $\mathcal{D}\left(\underline{r}\right)$ היא מילת הקוד המשוערכת. המפענח יודע כמובן מהו ספר הקוד, ולרוב יודע בדיוק באיזה ערוץ משתמשים (אם הערוץ לא ממש ידוע, אז יש דרכים כדי ללמוד אותו). לכן, המפענח יכול להשתמש במידע על הערוץ, כמו למשל הסתברות המעבר $\Pr(\cdot|\cdot)$. הסתברות שגיאת הפיענוח הממוצעת של מפענח כלשהו $\mathcal{D}$ נתונה על־ידי

$$
\begin{aligned}
P_{\text{err}} &\triangleq \sum_{\underline{x}\in\mathcal{C}} \Pr(\text{err}|\underline{c}) \Pr(\underline{c}),\\
\Pr(\text{err}|\underline{c}) &= \sum_{\substack{\underline{r}\in\mathcal{Y}^n\\ \mathcal{D}(\underline{r})\neq\underline{c}}} \Pr(\underline{r}|\underline{c}).
\end{aligned}
$$

המפענח שמביא למינימום את הסתברות השגיאה הממוצעת נקרא מפענח MAP – Maximum A-Posterior והוא נתון על־ידי

$$\mathcal{D}_{MAP}(\underline{r}) = \arg\max_{\underline{c}\in\mathcal{C}} \Pr(\underline{c}|\underline{r}).$$

במילים אחרות, המפענח הטוב ביותר שניתן למצוא (מבחינת מזעור הסתברות השגיאה הממוצעת) עובר על כל מילות הקוד ובוחר בזאת שהסיכוי שהיא נשלחה (בהינתן הווקטור הנקלט $\underline{r}$) היא הגבוהה ביותר. במקרה שמילות הקוד נשלחות בהסתברות שווה (ורק במקרה זה) מפענח MAP זה למפענח ML – Maximum Likelihood

$$\mathcal{D}_{ML}(\underline{r}) = \arg\max_{\underline{c}\in\mathcal{C}} \Pr(\underline{r}|\underline{c}).$$

היתרון במפענח ML הינו בכך ש־$\Pr(\underline{r}|\underline{c})$ זה בעצם הסתברות המעבר של הערוץ, עובדה שמובילה לפישוט מסוים.

דוגמה 14 בערוץ הבינארי הסימטרי $BSC(p)$ חסר־הזכרון עם הפרמטר $p \in [0, \frac{1}{2})$ נקבל

$$
\begin{aligned}
\mathcal{D}_{ML}(\underline{r}) &= \arg\max_{\underline{c}\in\mathcal{C}} \Pr(\underline{r}|\underline{c})\\
&= \arg\max_{\underline{c}\in\mathcal{C}} \prod_{i=1}^{n} \Pr(r_i|c_i)\\
&= \arg\max_{\underline{c}\in\mathcal{C}} \prod_{i=1}^{n} p^{r_i\oplus c_i}(1-p)^{1-r_i\oplus c_i}\\
&= \arg\max_{\underline{c}\in\mathcal{C}} p^{\sum_{i=1}^{n} r_i\oplus c_i}(1-p)^{n-\sum_{i=1}^{n} r_i\oplus c_i}\\
&= \arg\max_{\underline{c}\in\mathcal{C}} \left(\frac{p}{1-p}\right)^{d_H(\underline{c},\underline{r})}\\
&\overset{p<\frac{1}{2}}{=} \arg\min_{\underline{c}\in\mathcal{C}} d_H(\underline{c},\underline{r}).
\end{aligned}
$$

כלומר, עבור הערוץ הבינארי הסימטרי $BSC(p)$ חסר־הזכרון עם הפרמטר $p \in [0, \frac{1}{2})$, פיענוח ML שקול למציאת מילת קוד בעלת מרחק Hamming מינימלי מהווקטור הנקלט.

מהקשר הזה בין פיענוח ML לבין מרחק Hamming נסיק שכמה שהקוד הוא בעל **מרחק קוד** גדול יותר, כך יכולות התיקון שלו יותר טובים.

משפט 4 קוד עם מרחק d יכול לתקן בוודאות $\lfloor\frac{d-1}{2}\rfloor$ שגיאות, ו־ $d-1$ מחיקות.

עוד לא אמרנו **איך** מתקנים את השגיאות/מחיקות מבחינה פרקטית, אבל משפט 4 קובע ש**ניתן** לעשות זאת. ממשפט 4 ניתן להסיק ש: 1) עדיף קוד עם מרחק כמה שיותר גדול, 2) ניתן לתקן יותר מחיקות מאשר שגיאות. המסקנה השניה הגיונית, שכן בתיקון מחיקות יש לנו מידע נוסף שהוא מיקומי המחיקות (האינדקסים שנמחקו), ואילו בתיקון שגיאות אנחנו לא יודעים היכן הן נפלו.

מצד שני, הגדלת מרחק הקוד d בדר"כ מקטינה את גודל הקוד M (ראו משפט 3) ולכן מקטינה גם את קצב הקוד R. האתגר הוא למצוא קודים עם מרחק גדול וקצב גבוה, אשר ניתנים לפיענוח בסיבוכיות יעילה!

דוגמה 15 בערוץ הגאוסי AWGN(σ) חסר הזיכרון, עם מיפוי BPSK – $'0'\to +A, '1'\to -A$ נקבל

$$\begin{aligned}
\mathcal{D}_{ML}(\underline{r}) &= \arg\max_{\underline{c}\in\mathcal{C}} \Pr(\underline{r}|\underline{c}) \\
&= \arg\max_{\underline{c}\in\mathcal{C}} \prod_{i=1}^{n} \Pr(r_i|c_i) \\
&= \arg\max_{\underline{c}\in\mathcal{C}} \prod_{i=1}^{n} \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(r_i-c_i)^2}{2\sigma^2}} \\
&= \arg\max_{\underline{c}\in\mathcal{C}} e^{-\sum_{i=1}^{n}\frac{(r_i-c_i)^2}{2\sigma^2}} \\
&= \arg\min_{\underline{c}\in\mathcal{C}} \sum_{i=1}^{n} (r_i-c_i)^2 \\
&= \arg\min_{\underline{c}\in\mathcal{C}} \sum_{i=1}^{n} r_i^2 - 2r_ic_i + A^2 \\
&= \arg\max_{\underline{c}\in\mathcal{C}} \sum_{i=1}^{n} r_ic_i\,.
\end{aligned}$$

כלומר, פעינוח ML שקול למציאת מילת קוד בעלת קורלציה מירבית עם אות המוצא. לדוגמא, אם משתמשים בקוד החזרות שבו $\mathcal{C}=\{(+A,+A,\ldots,+A),(-A,-A,\ldots,-A)\}$ אז מהביטוי לעיל נקבל שפיענוח ML בערוץ הגאוסי שקול ל־

$$\mathcal{D}_{ML,rep}(\underline{r}) = \begin{cases} (+A,+A,\ldots,+A) & \sum_{i=1}^{n} r_i > 0 \\ (-A,-A,\ldots,-A) & \sum_{i=1}^{n} r_i < 0 \end{cases}.$$

הפיענוח לעיל בערוץ הגאוסי, שבו משתמשים באות המוצא הרציף של הערוץ ללא שינויו נקרא Soft Decision. אופציה אחרת, הינה פיענוח מסוג Hard Decision שבו קודם ממירים את אות המוצא לערכים בדידים (דה־מודולציה) ואז מפעילים מפענח קשיח מתאים. למשל, אופציה אחת הינה לשער שאם r_i חיובי, אז שודר $+A$, כלומר ביט שערכו $'0'$, ואם r_i שלילי, אז שודר $-A$, כלומר ביט שערכו $'1'$. רצף הפעולות הללו (מיפוי BPSK, ערוץ גאוסי, ודה־מודולציה) שקולים למעשה לערוץ BSC, ולכן ניתן כעת להפעיל מפענח קשיח מתאים.

מכיוון שבפעולת הדה־מודולציה איבדנו מידע, אז פיענוח Hard Decision בהגדרה נותן הסתברות שגיאה גדולה יותר מאשר פיענוח Soft Decision.

למרות הפישוטים האפשריים, מפענח ML הוא בעל סיבוכיות גבוהה מדי, מכיוון שצריך לעבור על **כל מילות הקוד** (ראו דוגמא 14). עבור קצב קבוע R נקבל שמספר מילות הקוד M גדל בצורה אקספוננציאלית באורך הקוד ($M = 2^k = 2^{nR}$), וכדי לקבל קודים טובים צריך להשתמש באורכים גדולים. למשל, בקוד באורך $1\text{KB} = 8192\text{b}$ (סדרי גודל בזכרונות Flash) וקצב $R = \frac{1}{2}$ יש $10^{1233} \approx 2^{4096}$ מילות קוד! עם זאת, בעזרת תכנון חכם של קוד בעל מבנה מסוים, ייתכן וניתן לממש מפענחים טובים בסיבוכיות שהיא פולינומיאלית באורך הקוד, כמו למשל בקודי RS, BCH. יתרה מכך, בעזרת פיענוחים תת־אופטימלים, ניתן אפילו לקבל סיבוכיות פיענוח שהיא לינארית או לינארית לוגריתמית באורך הקוד, כמו למשל בקודי LDPC, Polar .

7 קודי גרף ופיענוח איטרטיבי בערוץ המחיקה

בפרק זה נסקור בצורה חלקית משפחת קודים נפוצה שנקראת קודי LDPC – Low-Density Parity-Check. מה שמושך מבחינה פרקטית במשפחת קודים זו היא אלגוריתם הפיענוח המהיר שלה שנקרא Belief Propagation (BP). קודי LDPC נבחרו למספר רב של תקני תקשורת (WiFi, 5G, Deep-Space Communications) כמו גם בהתקני זכרון (SSD, Flash).

הערה 3 אנחנו נגביל את העיסוק לקודי LDPC בינארים בלבד. קיימות גם גרסאות מעל שדות גדולים יותר, אך לא נעסוק בהם כאן.

הערה 4 אנחנו נתמקד באלגוריתם פיענוח על גבי ערוץ המחיקה הבינארי (BEC) בלבד. קיימת גרסאות גם לערוצים בינארים־במבוא וחסרי־זכרון כלליים (כמו למשל, BSC, AWGN, Laplace, etc.), אך למען הפשטות לא נכנס לזה כאן.

הצגה ומוטיבציה

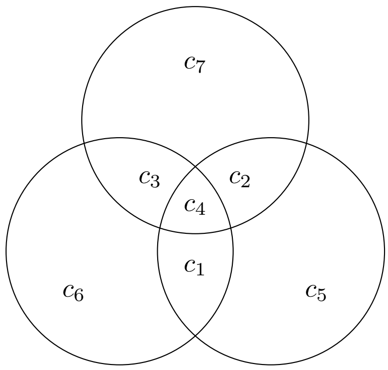
נניח קוד ליניארי בינארי באורך $n = 7$ עם מטריצת בדיקת זוגיות (ראו פרק 5) מהצורה הבאה:

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

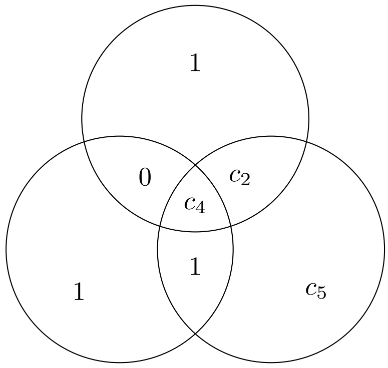
מכיוון שלכל מילת קוד $\underline{c} = (c_1, c_2, \ldots, c_7)$ צריך להתקיים $Hc^T = 0$, אז ניתן לכתוב

$$\begin{array}{ccccccccccccccc} c_1 & + & c_2 & + & & + & c_4 & + & c_5 & & & & & = 0 \\ c_1 & + & & + & c_3 & + & c_4 & + & & & c_6 & & & = 0 \\ & & c_2 & + & c_3 & + & c_4 & + & & & & & c_7 & = 0 \end{array}$$

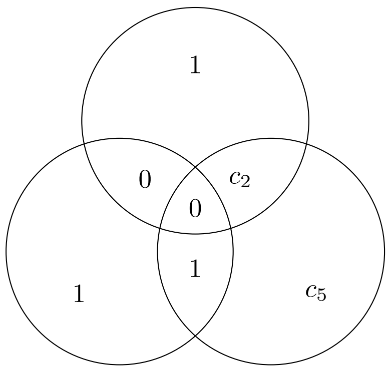
כאשר כל החיבורים הם XOR. את משוואות אילוצי הזוגיות האלו ניתן להציג בצורה גרפית כך שכל אילוץ הינו עיגול המקיף את המשתנים שמשתתפים בו. הצגה זו נקראת דיאגרמת Venn. עבור הדוגמא לעיל נקבל איור כזה



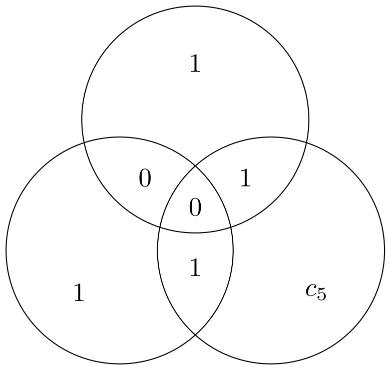
כעת נניח שידור של מילת קוד על גבי ערוץ המחיקה הבינארי, ונניח שהתקבל האות $(1, ?, 0, ?, ?, 1, 1)$. האיור שלנו נראה כך:



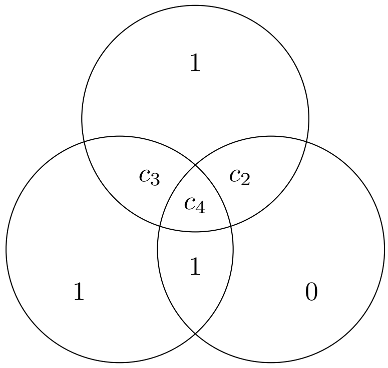
השאלה היא כמובן, מהם הערכים של האינדקסים המחוקים c_2, c_4, c_5? הפיענוח מתבצע בצורה איטרטיבית, כאשר בכל איטרציה מנסים לפענח ביט יחיד בעזרת אילוץ זוגיות מסוים. למשל, בעזרת האילוץ השמאלי באיור ניתן להסיק ש־ $c_4 = 0$, אשר מוביל לאיור הבא:



כעת, בעזרת האילוץ העליון נסיק ש־ $c_2 = 1$, אשר מוביל לאיור הבא:



לבסוף, מהאילוץ הימני נקבל ש־ $c_5 = 0$. לסיכום, מילת הקוד ששודרה הינה $(1,1,0,0,0,1,1)$.
נבחן דוגמא נוספת, בה האות שמגיע מהערוץ הינו $(1,?,?,?,0,1,1)$, כלומר הדיאגרמה היא:



ניתן לראות שכעת המפענח תקוע. כל אילוץ זוגיות (עיגול) מכיל לפחות שתי מחיקות, ולכן לא ניתן להתחיל את תהליך הפיענוח. מצד שני, אנחנו יכולים להגיד ש־

$$\begin{array}{ccccccl} c_2 & + & & & c_4 & = 1 & \\ & + & c_3 & + & c_4 & = 0 & . \\ c_2 & + & c_3 & + & c_4 & = 1 & \end{array}$$

למערכת משוואות אלו יש פתרון יחיד: משתי המשוואות האחרונות **יחד** ניתן להסיק ש־ $c_2 = 1$, אשר יחד עם המשוואה הראשונה מוביל ל־ $c_4 = 0$, ולבסוף $c_3 = 0$.

מה קרה פה? איך בשיטת הפיענוח האיטרטיבי נתקענו, ואילו בשיטת הפיענוח השניה הצלחנו? התשובה היא שהפיענוח האיטרטיבי הינו תת־אופטימלי ואילו הפיענוח השני הוא אופטימלי (למעשה הוא מימוש של פיענוח ML). ישנם מקרים שבהם הפיענוח האיטרטיבי ייכשל בעוד שפיענוח ML לא ייכשל. כמובן שהכיוון השני איננו נכון: אם הפיענוח האיטרטיבי הצליח, אז גם פיענוח ML היה מצליח.

במה שונה הפיענוח האיטרטיבי מהפיענוח האופטימלי? ההבדל המהותי ביניהם הוא שבפיענוח האיטרטיבי אנחנו מפענחים כל אילוץ זוגיות בצורה מקומית, ללא תלות באילוצים אחרים, ואילו בפיענוח האופטימלי פתרנו את שלושת האילוצים יחד, כמערכת של שלוש משוואות בשלושה נעלמים. זה נראה כמו חדשות לא טובות; למה שנרצה להשתמש באלגוריתם תת־אופטימלי שכזה? למעשה, פה דווקא טמון היתרון הגדול שיש לפיענוח האיטרטיבי על־פני שיטות פיענוח אחרות. העובדה שפותרים כל אילוץ בנפרד – ולא כולם יחד – מביאה להקלה בסיבוכיות הפיענוח, ומאפשרת שידור של מילות קוד ארוכות מאוד (אפילו לסדרי גודל של מיליוני ביטים), בקצב גבוה, והסתברות שגיאה נמוכה. למעשה, פיענוח איטרטיבי היווה פריצת דרך משמעותית בכך שאיפשר תקשורת אמינה בערוץ הגאוסי בקצבים שקרובים לקיבול הערוץ (הקצב האולטימטיבי שמעליו לא ניתן לקיים תקשורת אמינה) עם פיענוח פרקטי, תחילה באמצעות קודי טורבו (Turbo Codes) ולאחר מכן עם קודי LDPC.

הגדרת אלגוריתם הפיענוח

הגדרה 10 מטריצה תיקרא דלילה (sparse) אם מספר האלמנטים בה גדל ליניארית במספר העמודות.

דוגמה 16 אם נגריל מטריצה H בגודל $\alpha \cdot n \times n$ עבור α חיובי כלשהו, כך שכל איבר במטריצה הינו '1' בהסתברות $p > 0$ כלשהי, אז מספר האחדות במטריצה יהיה בקירוב $p \cdot \alpha \cdot n^2$ כלומר, ריבועי במספר העמודות n. אז, המטריצה הזו **לא תהיה דלילה**.

דוגמה 17 אם נגריל מטריצה H בגודל $\alpha \cdot n \times n$ עבור α חיובי כלשהו, כך שבכל עמודה יש בדיוק l אלמטים שאינם אפס, אז המטריצה הזו תהיה דלילה.

הגדרה 11 קוד ליניארי ייקרא קוד LDPC אם קיימת עבורו מטריצה בודקת שהינה דלילה.

הדלילות של המטריצה הבודקת היא תנאי חשוב בבניית הקוד. היא מאפשרת פיענוח בסיבוכיות שגדלה לינארית עם אורך הבלוק, וגם מראה ביצועי פיענוח טובים (בהקשר של הסתברות השגיאה). נשאלת השאלה, איך נבנה את המטריצה הבודקת כך שתיהיה דלילה ושביצועי הפיענוח אליה יהיו טובים? הכלי המרכזי שאיתו מתכננים קודי LDPC ודרכו גם מבצעים אנליזה של הביצועים הינו גרף דו־צדדי.

הגדרה 12 גרף $\mathcal{G} = (V, E)$ הינו אוסף של צמתים V וקשתות $E \subseteq \{\{v, u\} : v \in V, u \in V\}$. גרף ייקרא דו־צדדי (Bipartite) אם יש בו שני סוגים של צמתים, כך שקשתות מחברות צמתים שהם אינם מאותו סוג בלבד. פורמלית, $\mathcal{G} = (V_1 \cup V_2, E)$, ו־ $E \subseteq \{\{v, u\} : v \in V_1, u \in V_2\}$.

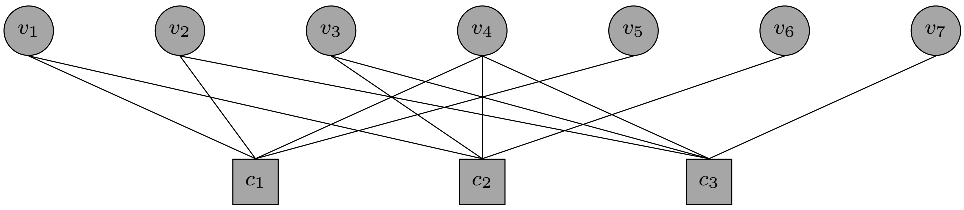
הגדרה 13 תהי H מטריצה בודקת של קוד לינארי. גרף טאנר (Tanner) של המטריצה הינו גרף דו־צדדי (לא מכוון) $\mathcal{G} = (V \cup C, E)$, כך ש־$V$ קבוצת צמתים המתארים את עמודות H (הביטים של מילת קוד), ו־C קבוצת צמתים המתארים את שורות H (את אילוצי הזוגיות). בגרף קיימת קשת $e = \{v, c\} \in V \times C$ אם ורק אם $H_{c,v} = 1$.

לקבוצת הצמתים V בגרף טאנר קוראים צמתי משתנה (Variable Nodes), ולקבוצת הצמתים C קוראים צמתי בדיקה (Check Nodes). נהוג לשרטט את גרף הטאנר כאשר צמתי המשתנה הם עיגולים, וצמתי הבדיקה הם ריבועים.

דוגמה 18 גרף הטאנר של המטריצה

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

נראה כך:



שיטת הפיענוח של קודי LDPC הינה אלגוריתם העברת הודעות. זה אומר שאחרי שמתקבל מידע מהערוץ, הצמתים מעבירים ביניהם הודעות באיטרציות כדי להבין מהי מילת הקוד ששודרה. כל אלגוריתם העברת הודעות מבוסס על מנגנון שבו צומת א' שולח הודעות לצומת ב' על־גבי קשת שמחברת ביניהם, על בסיס ההודעות שנכנסו לצומת א' באיטרציה קודמת וכלל עדכון מסוים. יש כל מיני אלגוריתמי העברת הודעות, אשר נבדלים בכללי העדכון שנמצאים בצמתים, והאופטימלי ביותר מביניהם נקרא Belief Propagation (BP). תיאור והצדקה של כללי העדכון באלגוריתם BP חורג מהמסגרת שלנו, אבל בערוץ המחיקה, קיים שינוי מסוים אשר מאפשר להגדיר אלגוריתם שקול ופשוט יותר. אלגוריתם העברת הודעות זה נקרא Peeling Decoder , ושמו ניתן לו מכיוון שהוא מקלף את המחיקות אחת אחת (כמו בצל) – בכל פעם הוא חושף מחיקה אחת ומסיר אותה מהגרף, עד אשר לא נשארות יותר צמתים בגרף. האלגוריתם פועל כך:

אלגוריתם הפיענוח Peeling Decoder

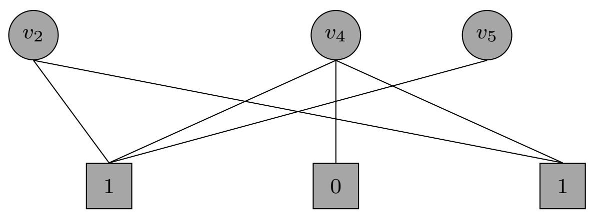
- **קלט:**
  - גרף טאנר $\mathcal{G} = (\{v_1, v_2, \dots, v_n\} \cup \{c_1, c_2, \dots, c_m\}, E)$
  - אות ממוצא ערוץ מחיקה בינארי $\underline{y} = (y_1, y_2, \dots, y_n) \in \{0, 1, ?\}^n$
- **פלט:** שיערוך $\underline{\hat{y}}_{PD} = (\hat{y}_1, \hat{y}_2, \dots, \hat{y}_n)$
- **אתחול:** קבע ערך 0 לכל צמתי הבדיקה.
- **צעדים:**

1. **לכל** ביט שאיננו מחוק y_i בצע:
   - $\hat{y}_i = y_i$
   - שלח את הערך של $\hat{y}_i$ לכל השכנים של צומת משתנה v_i והסר את v_i מהגרף $\mathcal{G}$ יחד עם כל הקשתות שלו.
   - בכל צומת בדיקה שמקבל הודעה, בצע XOR של ההודעה הנכנסת עם הערך הנוכחי של הצומת ועדכן את הערך בצומת
2. **כל עוד** יש צומת בדיקה c_j בגרף שהוא מדרגה 1 (רק קשת אחת מחוברת), בצע:
   - שלח את הערך של c_j לשכן (היחיד) שלו v_i, וקבע $\hat{y}_i = v_i$
   - שלח את הערך של $\hat{y}_i$ לכל השכנים של צומת משתנה v_i והסר את v_i מהגרף $\mathcal{G}$ יחד עם כל הקשתות שלו

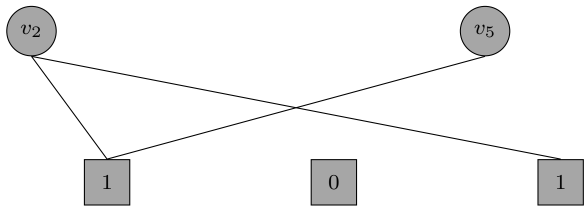
דוגמה 19 נבחן את התקדמות האלגוריתם עם הדוגמא מלמעלה:

$$H = \begin{pmatrix} 1 & 1 & 0 & 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 1 & 0 & 1 & 0 \\ 0 & 1 & 1 & 1 & 0 & 0 & 1 \end{pmatrix} \qquad \underline{y} = (1, ?, 0, ?, ?, 1, 1).$$

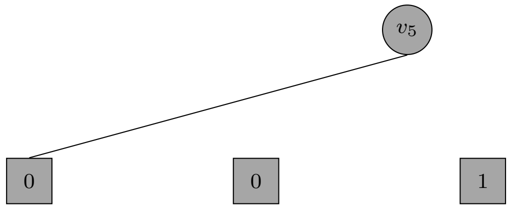
בשלב הראשון, האלגוריתם עובר על כל צמתי המשתנה שלא נמחקו, שולח את ערכם לצמתי הבדיקה ומסיר אותם מהגרף: צומת משתנה v_1 שולח את ההודעה 1 לצמתי הבדיקה השכנים שלו c_1, c_2 אשר מבצעים XOR עם הערכים הנוכחיים שלהם (אותחלו ל־0), ולכן $c_1 = 1, c_2 = 1, c_3 = 0$, ו־ v_1 מוסר מהגרף. לאחר מכן v_3 שולח את ההודעה 0 לצמתי הבדיקה השכנים שלו c_1, c_3; כעת $c_1 = 1, c_2 = 1, c_3 = 0$, ו־ v_3 מוסר. הבא בתור v_6 שולח את ההודעה 1 לצומת הבדיקה השכן שלו c_2 אשר מבצע XOR עם הערך הנוכחי שלו (1); כעת $c_1 = 1, c_2 = 0, c_3 = 0$, ו־ v_6 מוסר מהגרף. לבסוף, v_7 שולח את ההודעה 1 לצומת הבדיקה השכן שלו c_3; כעת $c_1 = 1, c_2 = 0, c_3 = 1$, ו־ v_7 מוסר. מתקבל הגרף הבא:



כעת האלגוריתם שולח הודעה מצומת בדיקה c_2 כי הוא מדרגה 1. ההודעה שנשלחת היא ערך הצומת 0, והיא נשלחת לצומת משתנה v_4. זה גורם למפענח להכריז ש $v_4 = 0$ לשלוח את ערכו לצמתי הבדיקה שמחוברים אליו, ולהסיר אותו מהגרף. מתקבלת התמונה הבאה:



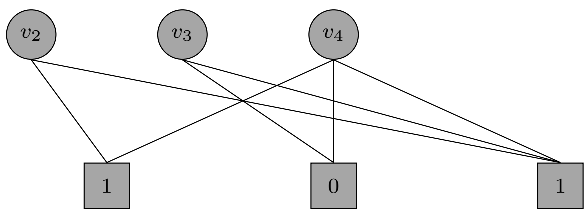
קיימת צומת בדיקה מדרגה 1 (צומת c_3), ערכה נשלח לצומת משתנה v_2 ולכן $v_2 = 1$. הצומת מוסר מהגרף והודעה נשלחת ל־c_1. מקבלים:



לבסוף, ערך המחיקה האחרון נחשף והלגוריתם מכריז על מילת הקוד $(1, 1, 0, 0, 0, 1, 1)$.

מתי אלגוריתם Peeling Decoder נכשל? מבדיקה בתיאור האלגוריתם, זה יקרה כאשר אין צמתי בדיקה בדרגה 1 בגרף, ויש עדיין מחיקות שעוד לא נחשפו.

דוגמה 20 בשימוש באותו הגרף כמו בדוגמא קודמת, אבל עם האות הנקלט $\underline{y} = (1, ?, ?, ?, 0, 1, 1)$ מקבלים אחרי ניפוי הביטים הידועים:

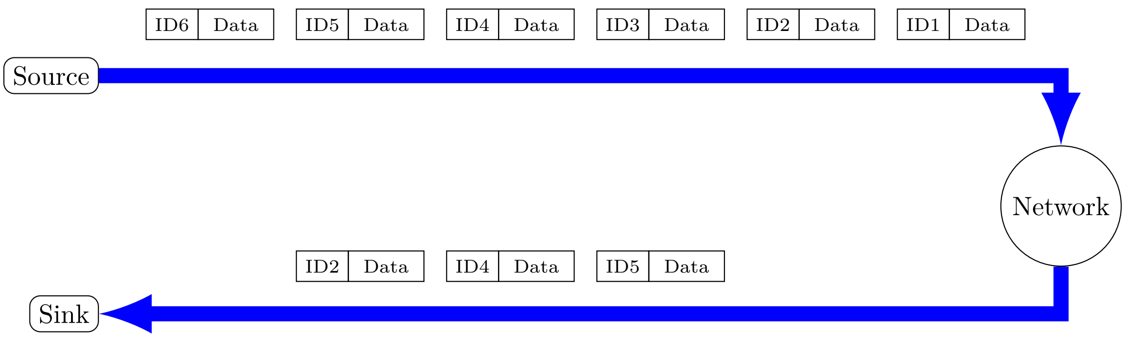


אין צמתי בדיקה מדרגה אחת, האלגוריתם נתקע ונכשל!

8 תיקון מחיקות ופרצים

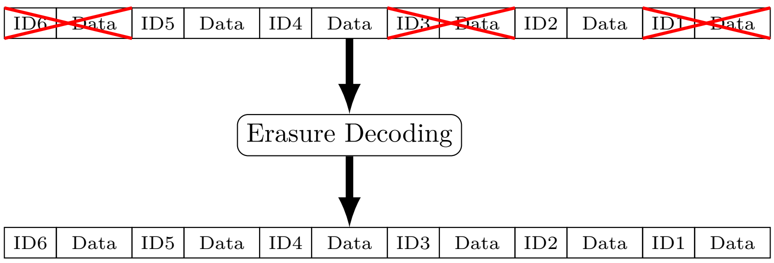
תיקון מחיקות שונה בעיקרו מתיקון שגיאות (אם כי לרוב משתמשים באותם הקודים), מכיוון שעל המפענח לדעת איפה בדיוק נפלו מחיקות! באפליקציות מסוימות, מידע זה זמין למפענח ולכן אפשר לתכנן את המערכת בהתאם. למשל, חשבו על חוות שרתים ענקית שמאוכסן בה מידע של חברת טכנולוגיה גדולה. לעיתים קרובות שרת כזה נופל והמידע בו אובד. מכיוון שידוע איזה שרת נפל, אז ניתן להתיחס עליו כאל מחיקה!

דוגמא נוספת היא פרוטוקול האינטרנט UDP – User Datagram Protocol, שבו חבילות packets עוברות במהירות גבוהה, על חשבון אמינות. לכל חבילה מוצמד מספר סידורי, וכך, אם חבילה אבדה בדרך, אז ניתן להתיחס על המידע בה כאל מחיקה. האיור הבא ממחיש את הבעיה



איור 5: נפילת חבילה באינטנט

אם חבילות 1 עד 6 באיור הנ"ל היו מילה בקוד שיכול לתקן את המחיקות האלו, אז היה ניתן לשחזר את כל החבילות:



איור 6: תיקון מחיקות

פרצים

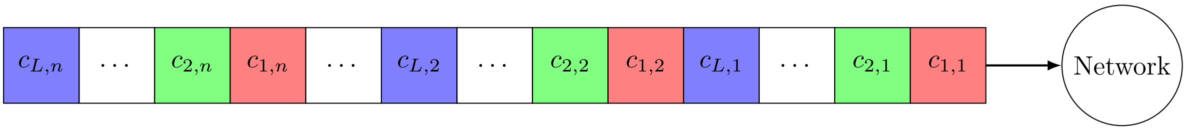
בתיאור לעיל של מערכת התקשורת הנחנו שמדי פעם נופלת חבילה, באופן בלתי תלוי בחבילה הקודמת. בשפה הפורמלית הנחנו שהערוץ הינו חסר־זכרון (ראו פרק 2). אולם, לעיתים הערוץ מכניס שגיאות/מחיקות בפרצים (bursts) ארוכים. למשל, קווי תקשורת חווים לפעמים נפילה שגורמת להם למחוק חבילות בפרצים. במצב כזה, הפתרון שנתנו בדמות קוד תיקון שגיאות/מחיקות איננו מספיק מכיוון שהפרץ עלול לשבש/למחוק מילות קוד שלמות! כדי להתגבר על בעיה זו משתמשים בשזירה של המידע (Data Interleaving). מטרת השזירה היא

לחלק את הפרץ באופן אחיד בין מילות קוד שונות, ובכך למנוע מצב שמילות קוד שלמות נמחקות. יש מספר שיטות לשזירה, ונתמקד בפשוטה מכולם (אך לא היעילה ביותר), שנקראת Block Interleaving. בשיטה זו, מכניסים L מילות קוד באורך n לתוך מערך מלבני בגודל $L \times n$ בצורה הבאה:

$$\begin{array}{|c|c|c|c|}\hline c_{1,1} & c_{1,2} & \cdots & c_{1,n} \\ \hline c_{2,1} & c_{2,2} & \cdots & c_{2,n} \\ \hline \vdots & \vdots & \ddots & \vdots \\ \hline c_{L,1} & c_{L,2} & \cdots & c_{L,n} \\ \hline \end{array}$$

איור 7: L מילות קוד באורך n בכניסה לשוזר

ומוציאים אותם לפי עמודות בצורה הבאה:



איור 8: L מילות קוד באורך n ביציאה מהשוזר

כעת פרץ של מחיקות/שגיאות ישפיע על מילות קוד שונות ולא ימחק מילות קוד שלמות. בצד המקבל מבצעים את הפעולה ההפוכה לשזירה ומקבלים בחזרה את מילות הקוד מיושרות. לאחר מכן מתבצע פיענוח שלהם והמידע מחולץ.

רשימת מקורות

[1] D. J. C. MacKay, “Information theory, inference & learning algorithms", *Cambridge University Press*, New York, NY, USA, 2002.

[2] S. Lin, and D. Costello, “Error control coding: fundamentals and applications", *Prentice-Hall*, Upper Saddle River, NJ, USA, 2004.

[3] T. K. Moon, “Error correction coding: mathematical methods and algorithms", *Wiley-Interscience*, New York, NY, USA, 2005.

[4] R. Roth, “Introduction to coding theory",*Cambridge University Press*, New York, NY, USA, 2006.

[5] T. Richardson and R. Urbanke, “Modern Coding Theory", *Cambridge University Press*, New York, NY, USA, 2008.