5. Десенко, С. М. Азагетероциклы на основе ароматических непредельных кетонов [Текст] / С. М. Десенко, В. Д. Орлов, П. Н. Гапоник, В. П. Каравай // Химия ачероцикл. соед. – 1990. – № 11. – С. 1533.

6. Дормидонтов, Ю. П. Методы УФ, ИК и ЯМР спектроскопии и их применение в органической химии [Текст]: учеб. пособ. / Ю. П. Дормидонтов. – Пермь: ПГУ химический факультет, 2001. – 79 с.

**References**

1. Krasovitsky, B. M., Lisova, I. V., Bogdanova, L. I. (1995). Research 1,3,3-trimethyl indoline derivatives oxazol-5-one. Ukr. chemical. Zh., 61 (7), 102–104.

2. Krasovitsky, B. M., Afanasiadi, L. M., Lisova, I. V., Stryukov, M. B., Lyubarskaya, A. E. (1982). Spectral-luminescent properties and electronic structure of some substituted azlactone and azlaktamov. Zh. chemistry, 61 (10), 2481–2485.

3. Krasovitsky, B. M., Bolotin, B. M. (1984). Organic luminophores. Moscow: Chemistry, 336.

4. Krasovitsky, B. M., Afanasiadi, L. M. (1997). Preparative Chemistry Organic luminophores. Kharkiv: Folio, 205.

5. Desenka, C. M., Orlov, V. D., Gaponik, P. N., Loaf, V. P. (1990). Azageterocikly na osnove aromatich-eskih nepredel'nyh ketonov. Himija acherocikl. coed., 11, 1533.

6. Dormidontov, Y. P. (2001). By UV, IR and NMR spectroscopy and its application in organic chemistry. Perm: PSU Department of Chemistry, 79.

*Рекомендовано до публікації д-р техн. наук Ведь В. Є.*
*Дата надходження рукопису 23.10.2015*

**Петров Сергей Александрович**, старший преподаватель, кафедра органического синтеза и нанотехнологий, Национальный технический университет «Харьковский Политехнический Институт», ул. Фрунзе, 21, г. Харьков, Украина, 61002
E-mail: petrowsa@gmail.com

**УДК 681.391**
**DOI: 10.15587/2313-8416.2015.53966**

# МЕТОДИ ОПТИМІЗАЦІЇ LDPC КОДУ З МЕТОЮ ПОКРАЩЕННЯ ПОРОГУ ВИПРАВЛЕННЯ ПОМИЛОК

**© Р. С. Новиков**

*Досліджено непорожні набори зупинок, які представляють собою головну причину досягнення порогового значення помилок в каналах передачі даних. Запропонований новий алгоритм перерахування найменших наборів зупинки і знаходження відстані зупинки для будь-якого коду LDPC. Запропоновано більш функціональна і гнучка техніка дроблення-і-заповнення. Розрахований час за який буде перераховано найменші набори зупинки і знайдено відстань зупинки коду LDPC*
***Ключові слова:*** *набір зупинки, завадостійке кодування, граф Таннера, матриця перевірок на парність*

*Non-empty stopping sets, which are the main reason for achieving a threshold of errors in data transmission channels, are studied. New algorithm of transfer smallest stopping sets and stop distance of any LDPC code is proposed. More functional and flexible technique of splitting-and-filling is proposed. Time for which will be transferred the smallest stopping sets and founded stop distance of any LDPC code is calculated.*
***Keywords:*** *stopping set, error control coding, Tanner graph, parity check matrix*

## 1. Вступ

Коди кінцевої довжини можуть досягати відмінних характеристик при низькій вартості обчислень при ітеративний передачі декодованих повідомлень. Тим не менш, LDPC код кінцевої довжини, особливо малих розмірів, неминуче досягає «порогу помилок». При використанні LDPC кодів в бінарних каналах були знайдені непорожні набори зупинки, які були головною причиною досягнення «порогу помилок».

Набори зупинки встановлені в матриці контролю парності коду LDPC відрізняються від наборів зупинки існуючих в матриці, що генерує LT код. Множина наборів зупинки спочатку не визначається і задається, якщо жорстке декодування припиняється без відновлення всіх джерельних вузлів. Тим не менш в коді LDPC, граф Таннеру в матриці контролю парності залишається незмінно з ітеративним декодуванням. Група перевірочних вузлів і складає набори зупинки, якщо жоден з їх сусідніх перевірочних вузлів не має ступінь один. Число цих перевірочних вузлів представляє собою розмір цього набору зупинки рис. 1.

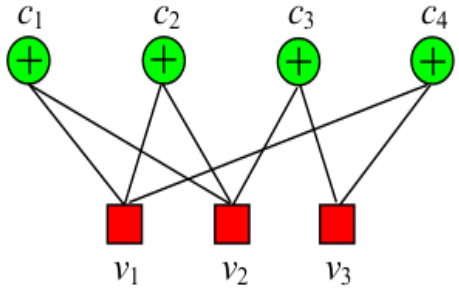



Рис. 1. Набір зупинок розміру 3 в матриці перевірок на парність

Непорожні набори зупинки – це набори зупинки, розмір яких перевищує 0. Найменший набір зупинок відіграє дуже важливу роль у коді LDPC на поріг помилки в бінарному каналі зі стиранням [1]. Розмір найменшого набору зупинок називається шлях зупинки (гальмівний шлях), або номер зупинки. Гальмівний шлях приблизно лінійний до довжини блоку коду LDPC, незалежно від того, це регулярний або нерегулярний код із заданим розподілом пари ступеню.

**2. Літературний огляд**

У літературних джерелах [1–3], були представлені корисні дослідження про взаємозв'язок між продуктивністю ансамблів LDPC коду і дистрибутиву набору зупинки, але автори не запропонували будь-які конкретні підходи для пошуку гальмівного шляху при використанні довільного LDPC коду. Причиною може бути властива NP-складність цих пошуків. У [4] був введений ефективний алгоритм пошуку наборів зупинки і наборів пасток (зупинок) для довільної, короткої довжини (≤500) LDPC коду, аналізуючи його верхню межу. Тим не менш, цей алгоритм дуже не зручний, оскільки довжина багатьох кодів в додатках більше, ніж 500. Дослідження [5, 6] містять деякі алгоритми пошуку або аналізу гальмівних наборів для індивідуального коду за допомогою комбінаторних або алгебраїчних методів, але ці підходи засновані на певних обмежених припущенях або додатках, тому вони не підходять для проектування загального LDPC коду. Це привело нас до дослідження нового підходу до пошуку гальмівного шляху і перерахування всіх маленьких наборів зупинки для будь-якого LDPC коду кінцевої довжини за допомогою деревовидного алгоритму пошуку в залежності від встановлених схем.

Метод дроблення-і-заповнення був введений Коу та ін. [7] для зменшення показників помилок коду LDPC. Їх метод, заснований на матриці репрезентації, може призвести до зниження щільності матриці контролю парності і розірвати деякі потенційно «погані» цикли. Ми запропонували більш функціональну і гнучку техніку, дроблення-і-заповнення (ДЗ), заснована на наборі діаграм (схем).

**3. Підхід до пошуку найменших наборів зупинки в LDPC коді**

1. Обхват коду LDPC, дорівнює принаймні 6 біт (пакетам).

2. В змінній (перевірочній) діаграмі набору вузлів, цикл – це закрита петля, утворена групою множин (наборів), серед яких будь-яка пара множин повинна розділяти один спільний елемент.

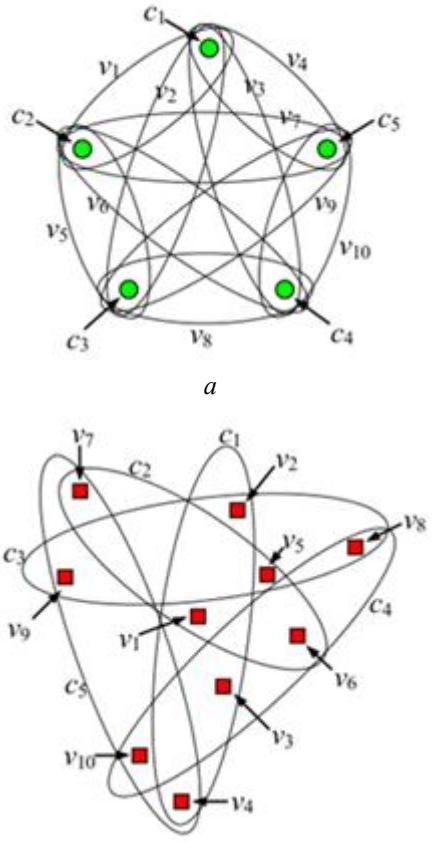
3. У наборі зупинки змінних вузлів, d (d>1) змінний вузол встановлює набір зупинки розміру d, якщо будь-який елемент міститься в об'єднанні цих d наборів змінних вузлів, то вони є елементами перетину щонайменше двох d змінних вузлів, де d – це гальмівний шлях або номер зупинки цієї діаграми набору.

Перевірочна матриця з блоком довжиною 9 задається як

$$H = \begin{array}{c} \begin{matrix} v_1 & v_2 & v_3 & v_4 & v_5 & v_6 & v_7 & v_8 & v_9 & v_{10} \end{matrix} \\ \begin{bmatrix} 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 1 & 1 \end{bmatrix} \end{array} \begin{matrix} c_1 \\ c_2 \\ c_3 \\ c_4 \\ c_5 \end{matrix} \quad (1)$$

.

Діаграми набору цього коду показані на рис. 2.




*а*

*б*

Рис. 2. Діаграми набору для матриці перевірок на парність **H**: *а* – діаграма набору змінного вузла; *б* – діаграма набору перевірочного вузла

В цьому LDPC коді з припущенням та визначенням циклу, ми бачимо, що цикл довжини *2d* саме d різний зв'язуваний набір перевірочних вузлів, що містять *d* різних елементів і кожен елемент містить чотири змінних набори вузлів (два набори перевірочних вузлів), щоб утворити цикл із *d* перевірочних вузлів. Визначення набору зупинки означає, що будь-який елемент в об'єднанні *d* змінних наборів вузлів містить щонайменше два з цих *d* змінних наборів вузлів. Іншими словами, існує *d* змінних наборів вузлів і щонайменше *d* різних елементів, кожен з яких містить будь-які два з змінних наборів вузлів в наборі зупинки з розміром *d*. Слід зазначити, що, тільки тоді, коли всі ці *d* різні змінні вузли мають ступінь 2 – число різних об'єднаних елементів дорівнює *d*. Має бути, принаймні один цикл в наборі зупинки [3].

Зробимо ще одне припущення, що в коді LDPC ступені всіх контрольних вузлів будуть не менше, ніж 2. Позначимо непорожній набір зупинок як *S*. У приймачі в bit error channel (BEC), ми припускаємо, що всі змінні вузли, що містяться в *S* є підмножиною стертих змінних вузлів. Декодер зупинить ітераційну процедуру декодування, коли всі передані біти відновляться або максимальна кількість ітерацій декодування буде закінчена. Оскільки всі елементи перевірочних вузлів в S покриті, принаймні, двома наборами змінних вузлів, всі декодовані повідомлення, які вони отримують від сусідніх з ними змінних вузлів – це невизначеності і значення цих елементів не можуть бути ідентифіковані. Таким чином, повернені від контрольних вузлів повідомлення завжди будуть невизначеними і змінні вузли в *S* не можуть бути відновлені, тому поріг помилок буде рости.

Виходячи з двох визначень і двох припущень, був запропонований новий алгоритм перерахування найменших наборів зупинки і знаходження відстані зупинки для будь-якого індивідуального коду LDPC з відстанню зупинки менше ніж 8 біт (пакетів). Цей алгоритм проводить деревовидний пошук в наборі діаграм перевірочної матриці LDPC. Пошукова діаграма базується в одному змінному вузлі і розгалужується від змінного вузла до змінного вузла з'єднані ребрами. Кожне ребро має позначку з групою контрольних вузлів, які з'єднані з можливими наборами зупинки. Шлях (пошуковий напрям) показує набір зупинки, якщо цей шлях успішний. У пошуковій діаграмі, змінний вузол в пошуковому кроці *j* називається стан при рівні *j*. Корінь (0-й ступінь) діаграми – це початковий стан, що є початком пошуку. Для зручності, термін стан ($s_j$), змінний вузол ($v_i$), і вершина використовуються сумісно для цього алгоритму (*i* – індекс змінного вузла). Ця пошукова схема, приводиться в дію двома функціями: функція виводу і функції перехідного стану пошуку.

Вихідна функція

$$O_j = \begin{cases} f_j(s_j), & j = 0, \\ f_j(s_j, I_{j-1}), & j > 0, \end{cases} \qquad (2)$$

де $O_j$ включає $I_j$, у пошуковому шляху на рівні *j*. І перехідна функція пошукового стану формулюється

$$s_j = \begin{cases} v_i, & j = 0, \\ g_j(s_{j-1}, I_{j-1},) & j > 0. \end{cases} \qquad (3)$$

Вихід $O_j$ – це об'єднання $I_j$ і сусідів вершини *i* ($v_i$). Серед вхідних параметрів вихідної функції, $S_J$ – це даний змінний вузол на поточному рівні *j* і $I_{j-1}$ – це поєднання контрольних вузлів, вироблених на попередньому рівні. $I_{j-1}$ складається з одного спеціального перевірочного вузла і кількох загальних контрольних вузлів. Спеціальний контрольний вузол включений в $I_{j-1}$ перевірочного вузла, який використовується для отримання поточного стану $S_j$, а решта загальні перевірочні вузли. На рівні *j*, якщо перевірочний вузол покритий як обидва і $I_j$ і сусідній набір стану $S_j$, цей перевірочний вузол повинен бути «обведеним» зеленим квадратом. Так «обведений» перевірочний вузол був відвіданий двічі або більше, тобто, він має принаймні двох сусідів.

Для того, щоб зменшити складність обчислень, пошуковий шлях, який повторюється на пошуковій діаграмі слід позначити як повторювальний. Правила зупинки у пошуковому шляху:

*Правило 1*: Якщо вершина в стані *x* така ж, як інша вершина в раніше обчисленому стані *y* і ці дві вершини призвели до того ж шляху пошуку, шлях подорожі від початкової вершини до цієї вершини буде дублікатом або недійсним, для його позначення використовується прапор провалу F для припинення пошуку в цій гілці. *x* і *y* надмірні (необов'язкові) на тому ж рівні пошуку.

*Правило 2*: Якщо всі контрольні вузли у виході $I_j$ по вершині «обведені», то пошук з початкової вершини до цієї вершини буде успішним і цей набір змінних вузлів включає набір зупинки. Ця успішна пошукова гілка відзначена прапором T, як кінцевий стан. Одночасно з цим всі інші гілки, в тому числі будь-яка з вершин в успішній гілці, встановлюються як ті, що не використовуються, та відзначаються прапором збою F і припиняються.

З пошуком функції переходу між станами, поточний стан $S_{J-1}$ і спеціальний перевірочний вузол генерують наступний стан $S_j$ відповідно до діаграми перевірочного вузлу. Наступний стан $S_J$ насправді може бути будь-яким з сусідів спеціального перевірочного вузла, крім поточного стану з $S_{J-1}$.

Як приклад, на рис. 3 показано пошук наборів зупинки, що покривають змінний вузол $v_1$ в коді LDPC представлений в матриці контролю парності, наведеною в рівнянні 1. Процес починається з $v_1$, який має двох сусідів ($c_1$ і $c_2$) і, отже, виробляє дві пошукові гілки на рівні 0. Для обох гілок "$c_1$, $c_2$" в результаті утворюються групи перевірочного вузла. На верхній гілці, $c_1$ спеціальний перевірочний вузол зверху позначений знаком "^", пошукові вершини на наступному рівні будуть серед сусідів $c_1$, а потім загальна перевірка вузла $c_2$. З іншого боку, $c_2$ є спеціальним перевірочним вузлом і $c_1$ загальний перевірочний вузол на нижній гілці. Верхня гілка йде на наступних рівень (стан), починаючи з рівня 1, які є сусідами $c_1$, крім $v_1$: $v_2$, $v_3$ і $v_4$. Для вихідної функції на $v_2$, вхідними параметрами є: $v_2$ ($s_1$) і "$c_1$, $c_2$" ($I_0$), так що вихідний $O_1$ буде об'єднанням $I_0$ і всіх сусідів $v_2$, тобто, $I_1 = \{c_1, c_2, c_3\}$. Тому що зараз $c_1$ був перекритий $I_0$ і сусідами $v_2$, він «обведений» зеленим полем. В цей час, $c_3$ стає спеціальним перевірочним вузлом для виведення пошуку на наступний рівень. Таким чином, $v_5$, $v_8$ і $v_9$ є стани (вершини) на рівні 2. Для $v_5$, його вихідна функція виробляє $I_2=\{c_1, c_2, c_3\}$ і, таким чином, всі ці перевірочні вузли «обведені» в зелені поля. Таким чином знаходиться набір зупинки – це об'єднання $v_1$, $v_2$ і $v_5$, які являють собою також успішний шлях пошуку "$v_1$, $v_2$, $v_5$". Для $v_8$, вихідні значення включають $I_2=\{c_1, c_2, c_3, c_4\}$, але тільки $c_1$ і $c_3$ «обведені» в зелені поля і $c_4$ спеціальний перевірочний вузол, для продовження пошуку. Крім того, вихідні данні для $v_9$ це $I_2=\{c_1, c_2, c_3, c_5\}$, і він не існує ні в одному з раніше обчислених шляхів, так що як і раніше необхідно продовжувати пошук, тому що тільки $c_1$ і

$c_3$ обведені. Повертаємося до рівня 1. Обидва $v_3$ і $v_4$ виводять нові шляхи пошуку, але тільки $c_1$ «обводиться», так що ці дві гілки розвивають нові стани на наступному рівні. На рівні 2, $v_6$ і $v_8$ видають різні дійсні шляхи. Тим не менш, $v_8$, $v_9$ і $v_{10}$, з'являються двічі на рівні 2, тому подальший шлях кожного припиняється. Аналогічно, для нижньої гілки на рівні 0, всі його стани не є успішними тому, що стани на рівні 1 вже існували в розвиненні верхньої гілки.

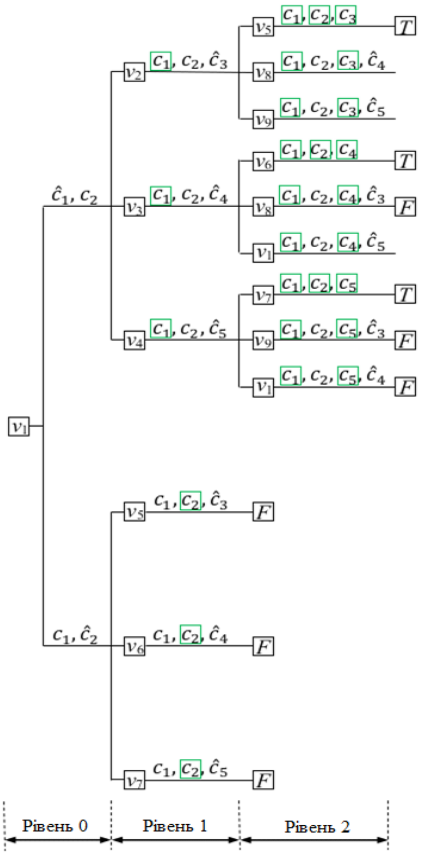



Рис. 3. Діаграма пошуку наборів зупинки

Пошуковий процес не продовжуватиметься на наступному рівні, якщо який-небудь набір зупинки був знайдений на поточному рівні. Після того, як всі змінні вузли при цьому алгоритмі пошуку закінчують пошук, всі найменші набори зупинки можуть бути знайдені, так шлях зупинки даного коду підтверджується. Для наведеного вище прикладу коду LDPC, всі множини найменших наборів зупинки – це {$v_1$, $v_2$, $v_5$}, {$v_1$, $v_3$, $v_6$}, {$v_1$, $v_4$, $v_7$}, {$v_2$, $v_3$, $v_8$}, {$v_2$, $v_4$, $v_9$} і {$v_3$, $v_4$, $v_{10}$}. Ці набори зупинки були обчислені за 0,228 секунд на комп'ютері 2,2-ГГц Intel CORE i5 CPU з 4 Гб оперативної пам'яті. У нашій моделі, алгоритм застосовується для двох випадково згенерованих кодів з довжиною блоків 1024 і 2304. Це зайняло 5380,328 секунд і 16615,21 секунд, відповідно, для того щоб обчислити всі найменші набори зупинки розміру 5.

Для перерахування всі найдрібніші набори зупинок, знаходимо $n$ пошукового дерева, n – кількість змінних вузлів в LDPC коді. На кожному дереві, обчислення складності в основному викликано кількістю успішних гілок і розміром найкоротших шляхів. Кількість дійсних гілок може бути оцінена змінним вузлом. Насправді, число дійсних гілок можуть бути пов'язані з $[(c_{avg}-1)(v_{avg}-1)]^{S_d}$, де $C_{avg}$ – середній ступінь перевірки вузла, $v_{avg}$ – середня ступінь змінного вузла і $s_d$ гальмівний шлях LDPC коду. Для будь-яких постійних або не постійних кодів LDPC кінцевої довжини n, доведено, що $s_d$ наближення до $\log\log(n)$. Розглянемо ступінь розподілу змінної вузла.

$$\lambda(x)=\sum_{i=1}^{n}\lambda_i x^i \qquad (4)$$

і ступінь розподілу перевірочного вузла

$$p(x)=\sum_{i=1}^{n-k}p_i x^i\ , \qquad (5)$$

де $\lambda_i(\rho_i)$ – ймовірність того, що змінний (перевірочний) вузол, займає, $i$-ту та $k$-ту ступінь коду виміру. Середня ступінь змінного вузла і середня ступінь перевірки вузлів може бути отриманий наступним чином:

$$v_{avg}=\lambda'(x)=\sum_{i=1}^{n}i\lambda_i x^{i-1} \qquad (6)$$

та

$$c_{avg}=\rho'(x)=\sum_{i=1}^{n-k}i\rho_i x^{i-1}\ . \qquad (7)$$

Ми також маємо

$$v_{avg}=(1-R)c_{avg}\ , \qquad (8)$$

де R – це швидкість коду.

Трудомісткість пошуку всіх найдрібніших наборів гальмування може бути представлено як $[(c_{avg}-1)(v_{avg}-1)]^{S_d}\times n$, в гіршому випадку може бути представлено як $[(c_{avg}-1)(v_{avg}-1)]^{\log(n)}\times n$. При гіршому випадку, вартість часу, головним чином, залежить від відстані гальмівного шляху і ступенів змінних і контрольних вузлів в коді LDPC [4].

Кількість маленьких наборів зупинок, як правило, набагато менше, ніж число гілок в поточному стані, споживчий простір для зберігання дійсних гілок кожного змінного вузла «домінує» над необхідним простором для процесу пошуку. Таким чином, він підходить для використання $O[(c_{avg}-1)\times$ $\times(v_{avg}-1)]^{S_d}$ простору.

### 4. Методика дроблення і заповнення

Основна ідея запропонованого методу дроблення-і-заповнення полягає в наступному: зі змінними (перевірочними) наборами схем, по-перше розділяється змінний (перевірочний) набір вузлів $S$ в бажану кількість змінних (перевірочних) наборів вузлів, а потім заповнює щойно створений змінний (перевірочний) набір вузлів елементами вихідного набору $S$, це робиться щоб задати їм задану ступінь розподілу і потрібні комбінаторні властивості. На стадії заповнення, немає необхідності використовувати всі

елементи у вихідному наборі вузла S [6]. Якщо всі ці елементи призначаються новому набору вузлів – процедура називається повним дробленням-і-заповненням рис. 4. В іншому випадку, це називається частковим дробленням-і-заповненням. Приклад часткового дробленням-і-заповненням проілюстровано на рис. 5. Набір змінних вузлів $v_i$ розділений на два змінних набори вузлів, $v_{i,1}$ і $v_{i,2}$ , але Елемент $c_g$ видаляється з заповнення.

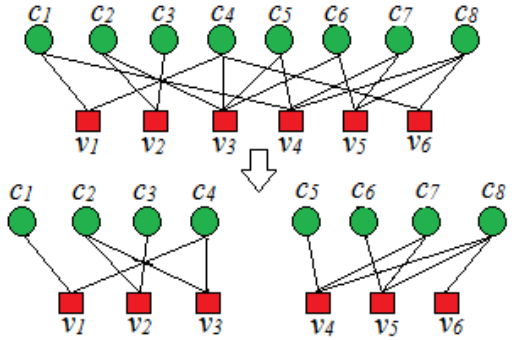



Рис. 4. Повне дроблення-і-заповнення у виді графу Таннера

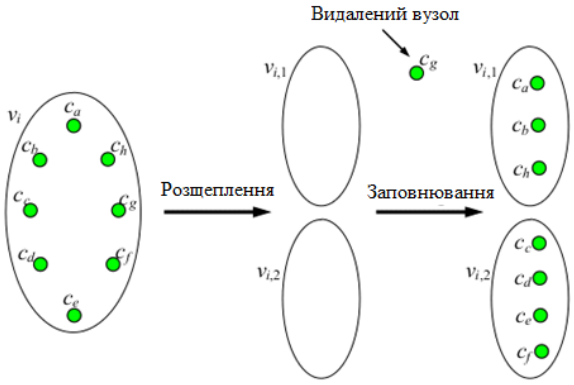



Рис. 5. Часткове дроблення-і-заповнення на наборі змінних вузлів

Для застосування техніки дроблення-і-заповнення, спочатку необхідно нарощувати схеми набору на основі евклідових геометрій. Потім, з правильно розробленою парою розділення ступеню, нерегулярний код LDPC заноситься, після застосування техніки дроблення-і-заповнення, в набір діаграм з деякими стратегіями. За допомогою навмисного заповнення в процесі генерації, можуть бути отримані специфічні комбінаторні характеристики коду.

Припустимо дається блок коду довжини $n$ та пара розподілу ступеня,тому може бути обчислена число ненульових входів в неправильній матриці контролю парності. В роботі [7] представлені рівняння для обчислення кількості одиниць в перевірочної матриці:

$$E = n \cdot \sum_{t \geq 2} \frac{(\lambda_i / i) i}{\int_0^1 \lambda(x) dx} = n \cdot \frac{1}{\int_0^1 \lambda(x) dx}. \qquad (9)$$

Далі представлено підхід для визначення вихідного набору діаграм. При такому підході необхідно знати параметри $E$ і $n$ для вибору правильної m-розмірності евклідової геометрії в $GF(2^S)$, що позначаються $EG(m, 2^S)$. m-розмірність Евклідової геометрії над $GF(2^S)$ може бути представлена в $2^{ms}$ m-наборів:

$$V = (b_0, b_1, \ldots, b_{m-1}), \qquad (10)$$

де $b_0, b_1, \ldots, b_{m-1}$ елементи $GF(2^S)$. Оскільки m-набір – це точка в геометрії Евкліда, наприклад $GF(m, 2^S)$, «пряма» лінії, яка проходить через точку $V_0$ може бути виражена

$$\{V_0 + \beta V : \beta \in GF(2^s)\}, \qquad (11)$$

де $V_0$ і $V$ лінійно незалежні в $EG(m, 2^S)$ і $V \neq 0$. Очевидно, що будь-які дві різні лінії мають не більше однієї загальної точки і будь-які дві точки з'єднані однією лінією

$$a = \frac{2^{ms} - 1}{2^s - 1} \qquad (12)$$

лінії, які проходять через кожну точку і кожен рядок мають $2^s$ точок в $EG(m, 2^S)$. Тепер змінний вузол і перевірочний набір діаграм можуть бути відображені в $EG(m, 2^S)$. Давайте розглянемо кожну точку як набор змінних вузлів і ліній, що перетинаються в цій точці, як її елементи (сусідніх контрольних вузлів). Крім того, кожна лінія розглядається як набір перевірочних вузлів і точки на лінії – її елементи. Таким чином, «сировина» регулярного набору діаграм (схем) може бути розроблена. Для того щоб розробити «сировину» набору схем, яка буде взаємодіяти з нашою технікою дроблення і заповнення, ми вибрали $m$ та $s$ параметри дуже обережно. Загалом, такі відносини повинні бути задоволені:

$$2^s \cdot \frac{2^{ms} - 1}{2^s - 1} \geq 2^E \qquad (13)$$

та

$$2^{ms} < n, \qquad (14)$$

де $n$ – довжина коду.

На наступному етапі, генерується неправильний код LDPC із заданим ступенем розподілу пари. Наступні три етапи переробки попереднього набору діаграм описані далі. На першому етапі, має бути обчисленим число змінних (перевірочних) вузлів з кожного призначеного ступеня. Рівняння розрахунку кількості змінних вузлів l ступеня і кількості контрольних вузлів до $k$:

$$N_{v,i} = n \cdot \frac{\lambda_i / i}{\int_0^1 \lambda(x) dx} \qquad (15)$$

та

$$N_{c,k} = n \cdot \frac{\rho_k / k}{\int_0^1 \rho(x) dx}. \qquad (16)$$

На другому етапі застосовується часткова техніку дроблення-і-заповнення. При правильному ре-

формування наборів вузлів, вона має усунути всі надлишкові ненульові елементи відповідної матриці контролю парності і підкоряється парі ступеня розподілу. Після цих операцій, генерується нова схема, $S_{G1}$ і всі набори в $S_{G1}$ повинні відповідати заданим ступеням. Третій крок виконує операцію повного дроблення-і-заповнення. На цьому етапі, інша діаграма, $S_{G2}$, відповідна $S_{G1}$ уточняється, щоб дані набори були наближені до бажаних ступенів і тримали всі ступені множин в $S_{G1}$ без змін.

**5. Результати досліджень**

За результатами, отриманими при зміні характеристик для наведених моделей, можна зробити висновок, що алгоритм знаходження найменших наборів зупинок дозволяє знайти набори зупинок за 0,228 секунд, для перевірочної матриці з блоком довжиною 9, та необхідно 5380,328 і 16615,21 секунд, для того щоб обчислити всі найменші набори зупинки розміру 5, для двох випадково згенерованих кодів з довжиною блоків 1024 і 2304, відповідно. Для перерахування всіх найдрібніших наборів зупинок, треба знайти $n$ пошукового дерева, де $n$ – кількість змінних вузлів в LDPC коді. На кожному дереві, обчислення складності в основному викликано кількістю успішних гілок і розміром найкоротших шляхів. Кількість дійсних гілок може бути оцінена змінним вузлом.

Основна ідея запропонованого методу дроблення-і-заповнення полягає в тому, що розділяється змінний (перевірочний) набір вузлів $S$ в бажану кількість змінних (перевірочних) наборів вузлів, а потім заповнює щойно створений змінний (перевірочний) набір вузлів елементами вихідного набору $S$, це робиться щоб задати їм задану ступінь розподілу і потрібні комбінаторні властивості. Техніка дроблення і заповнення дозволяє привести до зменшення показників помилок коду LDPC.

**6. Висновки**

Науковою новизною проведених досліджень є сформовані рекомендації щодо оптимізації параметрів LDPC кодів, а саме застосування техніки дроблення-і-заповнення, а також алгоритму знаходження найменших наборів зупинки і шляху зупинки.

За результатами, отриманими при зміні характеристик для наведених моделей, можна зробити висновок, що алгоритм знаходження найменших наборів зупинок дозволяє швидко знайти всі набори зупинок , що призводить до зменшення загального часу при передачі інформації з використанням завадостійкого кодування.

**Література**

1. Richardson, T. Error-floors of LDPC codes [Text] / T. Richardson. – Flarion Technologies Bedminster, N. J. – 2003. – P. 1426–1435. – Available at: http://www.ldpc-codes.com/papers/ErrorFloors.pdf

2. Milenkovic, O. Asymptotic spectra of trapping sets in regular and irregular LDPC code ensemble [Text] / O. Milenkovic, E. Soljanin, P. Whiting // IEEE Transactions on Information Theory. – 2007. – Vol. 53, Issue 1 – P. 39–55. doi: 10.1109/tit.2006.887060

3. Tian, T. Construction of irregular LDPC codes with low error floors. Vol. 5 [Text]: conference / T. Tian. C. Jones, J. Villasenor, R. D. Wesel. – Communication, Control and Computing. – 2003. – P. 3125–3129. doi: 10.1109/icc.2003.1203996

4. Chih-Chun, W. Exhausting Error-Prone Patterns in LDPC Codes [Text] / W. Chih-Chun, R. S. Kulkarni, H. V. Poor // IEEE Transactions on Information Theory. – 2006. – P. 46–70.

5. Li, H. Construction of irregular LDPC codes with low error floors [Text]: conference / H. Li, W. Huang and J. Dill. – Communication, Control and Computing. – 2010. – P. 1123–1128.

6. Richardson, T. Design of capacity-approaching iirregular low-density parity-check codes [Text] / T. J. Richardson, M. A. Shokrollahi, R. L. Urbanke // IEEE Transactions on Information Theory. – 2001. – Vol. 47, Issue 2. – P. 619–637. doi: 10.1109/18.910578

7. Kou, Y. Low-density parity-check codes based on finite geometries: a rediscovery and new results [Text] / Y. Kou, S. Lin, M. P. C. Fossorier // IEEE Transactions on Information Theory. – 2001. – Vol. 47, Issue 7. – P. 2711–2736.

**References**

1. Richardson, T. (2003). Error-floors of LDPC codes. Flarion Technologies Bedminster, N. J., 1426–1435. Available at: http://www.ldpc-codes.com/papers/ErrorFloors.pdf

2. Milenkovic, O., Soljanin, E., Whiting, P. (2007). Asymptotic Spectra of Trapping Sets in Regular and Irregular LDPC Code Ensembles. IEEE Transactions on Information Theory, 53 (1), 39–55. doi: 10.1109/tit.2006.887060

3. Tian, T., Jones, C., Villasenor, J., Wesel, R. D. (2003). Construction of irregular LDPC codes with low error floors. Vol. 5. Communication, Control and Computing, 3125–3129. doi: 10.1109/icc.2003.1203996

4. Chih-Chun, W., Kulkarni, R. S., Poor, H. V. (2006). Exhausting Error-Prone Patterns in LDPC Codes. IEEE Transactions on Information Theory, 46–70.

5. Li, H., Huang, W., Dill, J. (2010). Construction of irregular LDPC codes with low error floors. Communication, Control and Computing, 1123–1128.

6. Richardson, T. J., Shokrollahi, M. A., Urbanke, R. L. (2001). Design of capacity-approaching irregular low-density parity-check codes. IEEE Transactions on Information Theory, 47 (2), 619–637. doi: 10.1109/18.910578

7. Kou, Y., Lin, S., Fossorier, M. P. C. (2001). Low-density parity-check codes based on finite geometries: a rediscovery and new results. IEEE Transactions on Information Theory, 47 (7), 2711–2736.

*Рекомендовано до публікації д-р техн. наук Поповський В. В.*
*Дата надходження рукопису 14.10.2015*

**Новиков Роман Сергійович**, аспірант, кафедра телекомунікаційних систем, Харківський національний університет радіоелектроніки, пр. Леніна, 14, м. Харьков, Україна, 61166
E-mail: novik.r13@gmail.com