УДК 519.854

І. П. Градинар (Ужгородський нац. ун-т)

НАБЛИЖЕНІ АЛГОРИТМИ ЗНАХОДЖЕННЯ ПЕРЕШКОДОЗАХИЩЕНИХ КОДІВ, ЩО КОРЕГУЮТЬ ОДНУ ПОМИЛКУ В Z-КАНАЛІ

The problem of building largest error-correcting codes, by which a problem of information losing during its transferring or saving solved, is considered in the paper. To the task applied a range of existing approximate algorithms and their modifications. By one of the developed approximate algorithms obtained a new lower bound size of error-correcting codes, which correct one error on the Z-channel at $n = 13$.

У роботі розглядається питання побудови перешкодозахищених кодів максимального об'єму, за допомогою яких вирішується проблема втрати інформації при її передачі чи збереженні. До поставленої задачі застосовано ряд існуючих наближених алгоритмів, а також, їх модифікацій. Одним із розроблених наближених алгоритмів одержано нову нижню оцінку об'єму перешкодозахищених кодів, що коригують одну помилку в Z-каналі при $n = 13$.

На сьогодні є багато важливих задач, які є NP-складними. Вони вимагають більш детального вивчення для розробки та вдосконалення вже існуючих методів їх вирішення. Серед таких, актуальних в наш час задач, особливе місце займає задача знаходження перешкодозахищених кодів. Сучасні методи зберігання та передачі інформації за допомогою комп'ютерів та Інтернету призвели до виникнення необхідності збереження цілісності інформації при цих процесах. Одним з шляхів вирішення цієї проблеми і є знаходження перешкодозахищених кодів.

Перешкодозахищений код

Нехай B^n — множина n -вимірних векторів $x = (x_1, \ldots, x_n)$, компоненти яких приймають значення 0 або 1, де n — довжина коду ($|B^n| = 2^n$).

Означення 1. *Двійковим кодом C називається довільна підмножина B^n.*

Об'ємом коду будемо називати потужність цієї підмножини.

Нехай вектор $x^i \in B^n$ із-за помилок при передачі інформації може переходити в один з елементів довільної множини $R(x^i),\ i = 1,\ \ldots,\ 2^n$.

Означення 2. *Двійковий код $C \subset B^n$ називається перешкодозахищеним, якщо для $\forall x, y \in C,\ x \neq y$, виконується умова $(R(x) \cup x) \cap (R(y) \cup y) = \varnothing$.*

Виділяють такий клас перешкодозахищених кодів, як коди з мінімальною асиметричною відстанню $\Delta = 2$ для Z-каналу, які можуть коригувати один помилковий перехід 1 в 0. Розглянемо такі коди більш детальніше.

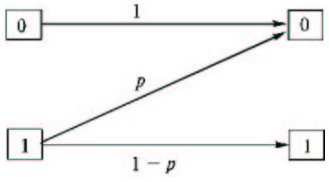

Рис. 1. Схема Z-каналу

Означення 3. *Z-каналом називається асиметричний двійковий канал, в якому при передачі інформації ймовірність помилки — переходу 1 в 0 рівна p, а ймовірність помилки — переходу 0 в 1, рівна нулю (рис. 1).*

Означення 4. *Асиметричною відстанню між векторами* $x, y \in B^n$ *називається величина*

$$d_A(x,y) = \max(N(x,y), N(y,x)),$$
$$\text{де } N(x,y) = \big|\{i | x_i = 0 \wedge y_i = 1\}\big|.$$

Мінімальна асиметрична відстань Δ для коду $C \subset B^n$ визначається наступним чином

$$\Delta = \min\{d_A(x,y) | x, y \in C, x \neq y\}.$$

Доведено, що код C, що має мінімальну асиметричну відстань Δ, може коректувати $\Delta - 1$ або менше асиметричних помилок, тобто переходів 1 в 0.

Ставиться задача знаходження перешкодозахищеного коду максимального об'єму для Z-каналу з мінімальною асиметричною відстанню $\Delta = 2$, тобто коду, що може коригувати одну помилку в Z-каналі.

В працях [1, 2] можна ознайомитися з нижніми оцінками об'єму перешкодозахищених кодів, що можуть коригувати одну помилку в Z-каналі при $n =$ 18–22, 24. Наприклад, при $n = 24$, знайдена в [1] нижня оцінка рівна 678 860. В даній роботі розглядаються тільки перешкодозахищені коди, що можуть коригувати одну помилку в Z-каналі при $n \leqslant 13$.

Зведення поставленої задачі до задачі на графі

Нехай заданий граф $G = (V, E)$, де $V = v_1, \ldots, v_k$ — множина вершин графу, $E = e_1, \ldots, e_m$ — множина його ребер ($k = |V|, \ m = |E|$).

Означення 5. *Підмножина* $I \subset V$ *вершин графу* G *називається незалежною, якщо ніякі дві його вершини не зв'язані ребром.*

Означення 6. *Незалежна множина* I_{max} *називається максимально незалежною множиною (MIS), якщо для будь-якої незалежної множини* I *виконується наступне співвідношення:* $|I_{max}| \geqslant |I|$.

Означення 7. *Незалежна множина* I *називається максимальною по включенню, якщо вона не є підмножиною деякої іншої незалежної множини.*

Будь-яка максимальна незалежна множина є максимальною по включенню, але не навпаки. Проблема полягає в знаходженні максимальної незалежної множини, тобто незалежної множини з максимальною кількістю вершин. Через $\alpha(G)$ прийнято позначати число вершин в максимально незалежній множині I_{max}. Тобто, $\alpha(G) = |I_{max}|$.

Відомо, що існують дві інші задачі на графах: пошук мінімального вершинного покриття та максимальної кліки. Всі три наведені задачі на графах мають широке застосування, є NP-складними і пов'язані між собою.

Задача відшукання максимально незалежної множини формулюється як задача чисельного програмування з булевими змінними (дискретна оптимізаційна задача). Кожній вершині $v_j \in V$ заданого графу $G(V, E)$ ставиться у відповідність змінна $x_j \in 0, 1, \ j = 1, \ \ldots, \ k$. Математична модель задачі відшукання максимальної незалежної множини наступна:

$$\max\left\{ f(x) = \sum_{i=1}^{k} x_i \prod_{j:(v_i,v_j)\in E} (1 - x_j) : x \in B^k \right\}.$$

Нехай

$$I(x) = \{v_j \in V : x_j = 1,\ j = 1,\ \ldots,\ k\},$$
$$\text{де } x = (x_1,\ \ldots,\ x_k).$$

Якщо виконується умова $f(x) = |I(x)|$, то $I(x)$ — незалежна множина на графі $G(V, E)$.

Задачу побудови перешкодозахищених кодів (включаючи коди, які корегують одну помилку в Z-каналі) можна звести до задачі знаходження максимальної незалежної множини вершин графу [2, 3]. Для цього граф $G(V, E)$ задається наступним чином. Нехай, $|V| = |B^n|$, тобто дорівнює 2^n. Встановлюється взаємно однозначне відображення $\chi : V \Rightarrow B^n$. Множина ребер

$$E = \{(v_i, v_j)|\ \ (R(\chi(x)) \cup \chi(x)) \cap$$
$$\cap\ (R(\chi(y)) \cup \chi(y))\ \neq \varnothing\}.$$

У графі $G(V, E)$ потрібно знайти незалежну множину I, відтак буде побудовано перешкодозахищений код, який визначається наступним чином: $C = \{\chi(v) : v \in I\}$.

Задачу отримання коду з мінімальною асиметричною відстанню Δ можна так само звести до задачі знаходження максимальної незалежної множини, лише при визначенні існування ребра між вершинами v_i та v_j можна використати умову $d_A(x_i, x_j) < \Delta$, де v_i та v_j — вершини, що відповідають векторам x_i та x_j відповідно. Тобто, якщо згадана умова виконується, то ребро між відповідними вершинами існує. Якщо знайдемо незалежну множину в такому графі, то це значить, що побудовано перешкодозахищений код, що корегує $\Delta - 1$ або менше асиметричних помилок. Він складається з двійкових векторів, що відповідають вершинам незалежної множини.

Таблиця 1.
К-ть вершин та ребер у lzc-графі

n	|V|	|E|
1	2	1
2	4	5
3	8	18
4	16	56
5	32	160
6	64	432
7	128	1 120
8	256	2 816
9	512	6 912
10	1 024	16 640
11	2 048	39 424
12	4 096	92 160
13	8 192	212 992
14	16 384	487 424
15	32 768	**1** 105 920
16	65 536	**2** 490 368
17	131 072	**5** 570 560
18	262 144	**12** 386 304
19	524 288	**27** 394 048
20	**1** 048 576	**60** 293 120
21	**2** 097 152	**132** 120 576
22	**4** 194 304	**288** 358 400
23	**8** 388 608	**627** 048 448
24	**16** 777 216	**1 358** 954 496
25	**33** 554 432	**2 936** 012 800
26	**67** 108 864	**6 325** 010 432

Оскільки нами розглядаються перешкодозахищені коди з асиметричною відстанню $\Delta = 2$, то при побудові відповідних графів необхідно розглядати умову $d_A(x_i, x_j) < 2$. Знаючи структуру таких графів (відомі як 1zc-графи), неважко вивести формулу кількості ребер в залежності від n, що й було зроблено в роботі:

$$|E| = \frac{1}{2} \times \sum_{k=0}^{n} \big(k + (n-k) + k(n-k)\big) \frac{n!}{k!\,(n-k)!},$$

де V — це множина вершин, E — множина ребер.

Кількість вершин та ребер у 1zc-графах для $n = \overline{1, 26}$ наведено в табл. 4. З неї видно, як із зростанням n зростає розмірність відповідного 1zc-графу.

Наближені алгоритми вирішення поставленої задачі

В працях [1–8] можна ознайомитися з наближеними алгоритмами, що застосовні до поставленої в роботі задачі. Методи, що використовують локальний пошук, займають провідне місце серед наближених методів розв'язання дискретних оптимізаційних задач. До таких відносять [3]: метод вектора спаду, табу, глобального рівноважного пошуку (GES) та ін.

Ідея локального пошуку пролягає в наступному. На початку задана деяка допустима множина G, цільова функція $f(x)$, а також початкова точка $x_0 \in G$ та її окіл $N(x_0)$. Вважається, що локальний оптимум знайдено, якщо окіл — не вироджена (не пуста) множина, і в ньому немає допустимої точки, що «покращує» значення цільової функції при переході з точки x_0 в цю точку. В іншому разі вибирається допустима точка з околу, що є «кращою» за x_0, і розглядається окіл вже відносно неї.

Процес продовжується до знаходження локального оптимуму.

Для алгоритмів локального пошуку необхідно [3] визначити наступне:

- систему околів;
- цільову функцію;
- стратегію побудови початкових рішень;
- стратегію переміщення.

Для задач з булевими змінними використовується відстань

$$\rho(x, y) = \sum_{j=0}^{n} |y_j - x_j|$$

між n-мірними булевими векторами x та y, і розглядається система околів точки $x \in G$ радіусу r виду

$$O(x, r) = \{y \in G : \rho(x, y) \leqslant r\},\ r = 1,\ 2,\ \ldots,\ n.$$

Як початковий розв'язок можна вибирати будь-яку допустиму точку $x_0 \in G$. Часто ж x_0 формують враховуючи особливості задачі, вже відомі локальні оптимуми та ін. Але і в такому випадку рекомендується [3] там, де це можливо, рандомізувати вибір початкових розв'язків.

Нехай перед нами ставиться задача дискретної оптимізації:

$$\min_{x \in G} f(x),$$

де G — допустима множина, $f(x)$ — цільова функція.

Наводять наступну узагальнену схему методів локального пошуку:

```
procedure локальний_пошук
⌈ початкові_налаштування(s, x^s)
| while пошук_покращення(x^s) ≠ «не існує» do
| ⌈ x^{s+1} = пошук_покращення(x^s)
| ⌊ s = s + 1
⌊ comment знайдено локальний мінімум x^s.
```

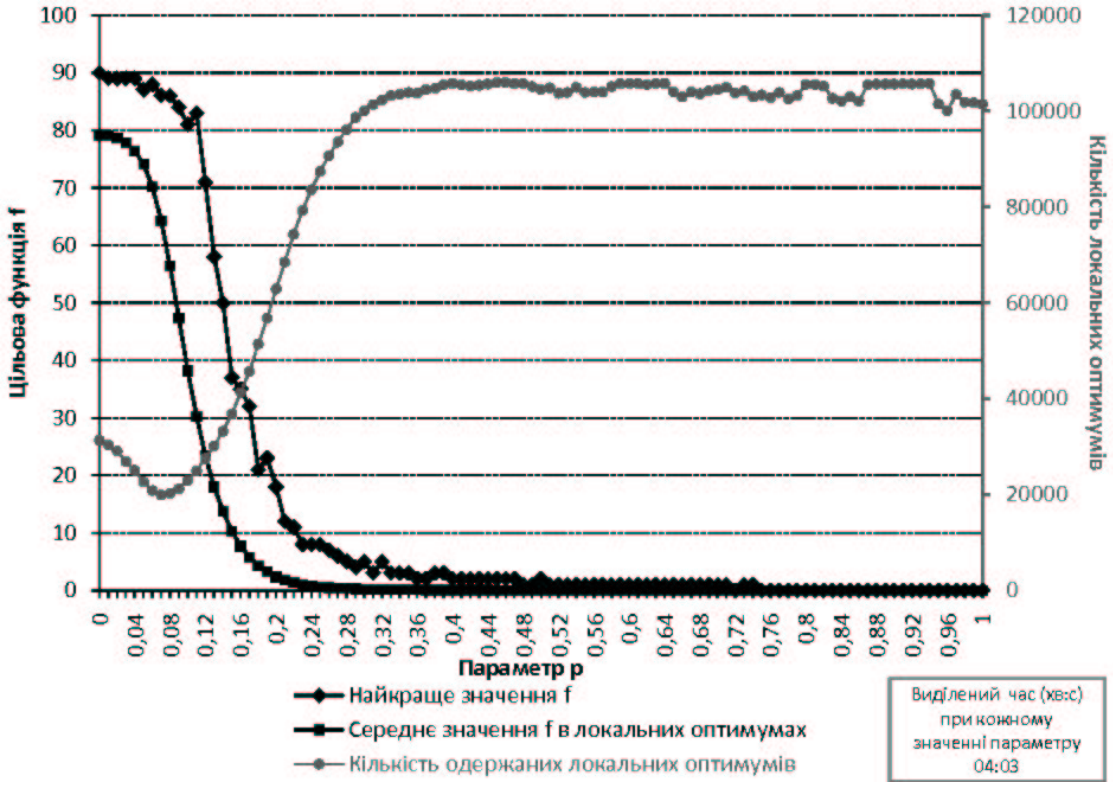

Рис. 2. Залежність від параметру p результатів одержаних методом вектора спаду ($r = 1$) при $n = 10$

Допоміжні процедури визначаються наступним чином:

procedure початкові_налаштування(s, x^s)

$\lceil$ кількість ітерацій $s = 0$;

$\lfloor$ вибір деякої початкової точки $x^0 \in G$;

$$\text{пошук\_покращення}(x^s) = \begin{cases} \text{будь-кому розв'язку } y \in N(x^s) : f(y) < f(x^s), \\ \text{якщо такий розв'язок існує;} \\ \text{«не існує» — в протилежному випадку;} \end{cases}$$

В залежності від стратегії переміщення, побудови початкових рішень, вибору системи околів, а також цільової функції, будемо мати різні методи локального пошуку. Інколи для пришвидшення роботи алгоритмів доцільно використовувати модифіковану цільову функцію.

В роботі було розглянуто локальний пошук на прикладі методу вектора спаду [3,5], вивчено його залежність від деяких параметрів (рис. 2, p - ймовірність, що x_i прийме значення 1, $\forall i = \overline{1,n}$).

При застосуванні методу вектора спаду до поставленої задачі, найкращі результати одержано коли початковий розв'язок був рівний $x_0 = (0, 0, \ldots, 0)$ і розглядалися околи радіуса $r = 1$. В іншому випадку за той же час одержували гірші результати. Тому, як зазначається в [3], доцільніше використовувати модифіковану цільову функцію. За допомогою методу вектора спаду одержано відомі нижні оцінки при $n \leqslant 7$. Результати наведено в табл. 2. Відомі нижні оцінки можна знайти в [1–4, 7–9].

Таблиця 2.
Нижні оцінки, одержані методом вектора спаду (p=0, r=1)

n	Виділений час (хв:с:мс)	Нижня оцінка		a	b	c	d	e
		Відома	Одержана					
4	00:00:333	**4**	4	65301	00:00:000	3,12	3	45
5	00:01:000	**6**	**6**	182676	00:00:000	4,98	8	260
6	00:03:000	**12**	**12**	4408	00:00:000	8,64	14	924
7	00:09:000	**18**	**18**	2357	00:00:000	14,79	25	3 247
8	00:27:000	**36**	33	12	00:00:953	25,50	49	12 527
9	01:21:000	**62**	55	4	00:08:937	44,71	89	45 480
10	04:03:000	**112**	92	2	02:34:656	79,12	148	151 972
11	12:09:000	**198**	158	4	04:34:797	141,28	233	477 450
12	36:27:000	**379**	278	1	05:29:984	254,10	358	1 465 429

a — к-ть повторень найкращого розв'язку, b — затрачений час на одержання першого найкращого розв'язку (хв:с:мс), c — середнє значення цільової функції в локальних оптимумах, d — к-ть одержаних локальних оптимумів, 10^5, e — к-ть ітерацій, 10^5

Розглядаючи поставлену задачу як задачу на графі, знаходження максимальної незалежної множини пропонується організувати шляхом вибору вершин найменшого степеня. Нехай на початку маємо деякий граф, кожній вершині якого відповідає її степінь, який легко можна визначити. Здійснюється рандомізований вибір вершини серед вершин найменшого степеня і добавляється у незалежну множину. З графу вона видаляється разом з околом, а степінь вершин з якими видалені вершини були інцидентні, зменшується на одиницю. Ці кроки повторюються кожен раз для новоутвореного графу до тих пір, поки не одержимо тривіальний або пустий граф. Тоді вершини, що залишились заносимо у незалежну множину. Після цього фіксуємо одержану потужність незалежної множини, і оскільки вибір вершин до незалежної множини здійснювався рандомізовано, процес запускаємо з початку, і можемо одержати іншу потужність. Знаходження максимальної незалежної множини шляхом послідовного вибору вершин найменшого степеня при $n \leqslant 9$ ($|V| \leqslant 512$, $|E| \leqslant 6912$), крім $n = 7$, дало відомі нижні оцінки (табл. 3).

Якщо при формуванні незалежної множини вибирати вершини найменшого та наступного за ним в порядку зростання степеня (назвемо такий вибір, як вибір вершин малого степеня), то, хоч для $n = 7$ і одержали відому нижню оцінку, але при $n \geqslant 11$ отримано менші нижні оцінки (табл. 3).

Таблиця 3.
Нижні оцінки, одержані шляхом вибору вершин найменшого (I) або малого (II) степеня

n	Назва графу, що відповідає	Виділений час (хв:с:мс)	Нижня оцінка		
			Одержана		Відома
			(I)	(II)	
4	1zc16	00:00:333	**4**	**4**	**4**
5	1zc32	00:01:000	**6**	**6**	**6**
6	1zc64	00:03:000	**12**	**12**	**12**
7	1zc128	00:09:000	16	**18**	**18**
8	1zc256	00:27:000	**36**	**36**	**36**
9	1zc512	01:21:000	**62**	**62**	**62**
10	1zc1024	04:03:000	108	108	**112**
11	1zc2048	12:09:000	188	186	**198**
12	1zc4096	36:27:000	340	334	**379**

Отже, знаходження максимальної незалежної множини шляхом вибору вершин найменшого степеня швидко дає досить велику потужність незалежної множини. Але незважаючи на рандомізований вибір вершин та виділення більше часу на багаторазовий запуск такого формування незалежної множини, не було досягнуто відомих нижніх оцінок для перешкодозахищених кодів, що коригують одну помилку в Z-каналі при $n \geqslant 10$. Виявлено деякі особливості такого формування незалежної множини, враховуючи які, можна визначити напрями вдосконалення. Вибір вершин найменшого степеня також легко можна застосувати разом з іншими наближеними алгоритмами, що може підвищити їх ефективність.

В [3,4] пропонується ефективний наближений алгоритм знаходження максимальної незалежної множини на графі. Загальна схема полягає в наступному.

Нехай на графі $G(V,\ E)$ ми маємо деяку максимальну по включенню незалежну множину $I \subset V$. Через V_j позначимо множину вершин $v_i \in V_j$, що зв'язані з вершиною $v_j \in V$. Очевидно, що множина $V_j \subset V$. Будемо вважати, що $v_j \in I$. Тоді для $\forall v_j \in I$, $\forall v_i \in V_j \quad v_i \notin I$. Через $List_j$ позначимо множину, що складається з тих вершин $v_i \in V_j$, в кон'юнктивних умовах

$$r_j = x_j \wedge \left[\bigwedge_{i:(v_i,v_j)\in E} \overline{x_i} \right]$$

яких саме 2 літерали мають значення фальш. Якщо такі вершини v_i існують, то v_j теж вважатимемо елементом множини $List_j$. В іншому разі множина $List_j = \varnothing$.

Через $List$ позначимо множину тих вершин $v_j \in I$, відповідна множина $List_j$ яких не є пустою. Отже, для незалежної множини $I \subset V$ визначено множини $List$ та $List_{j_q}$, $q = 1, \ldots, |List|$.

Лема 1. *Нехай $I(G(List_{j_q}))$ — довільна максимальна по включенню незалежна множина в графі $G(List_{j_q})$. Тоді множини вигляду $I\backslash\{v_{j_q}\} \cup I(G(List_{j_q}))$, $q = 1, \ldots, |List|$, є максимальними по включенню незалежними множинами в графі $G(V, E)$.*

Із справедливості леми слідує, що можна визначити окіл довільної максимальної по включенню незалежної множини I наступним чином:

$$O(I) = \left\{ I\backslash\{v_{j_q}\} \cup I(G(List_{j_q})),\ v_{j_q} \in List,\ q = 1, \ldots, |List| \right\}.$$

Для вирішення поставленої перед нами задачі можна використовувати цей окіл та застосовувати різні методи локального типу: метод вектора спаду, табу, глобального рівноважного пошуку (GES) та ін.

Крім того, відмічено, що основною проблемою є отримання максимальних по включенню множин $I(G(List_{j_q}))$. Оскільки, розмірність графу $G(List_{j_q})$ в основному невелика в порівнянні з розмірністю графу $G(V,\ E)$, а тому пошук множин $I(G(List_{j_q}))$ можна організувати методами локального типу радіуса $r = 1$ в просторі $B^{|List_{j_q}|}$ або точними методами.

Отже, в наведеному наближеному алгоритмі потрібно визначити:

1) спосіб одержання початкової, максимальної по включенню, незалежної множини I_0;

2) спосіб відшукання множин $I(G(List_{j_q}))$;
3) організацію пошуку локального оптимуму в околі $O(I)$;
4) критерій зупинки;

За алгоритмом, локальний оптимум буде знайдено, коли в околі $O(I)$ немає максимальних по включенню незалежних множин потужності більшої ніж $|I|$.

У випадку знаходження локального оптимуму алгоритм можна запустити знову. При цьому кожний раз певним чином повинна вибиратися інша початкова множина I_0, а найкращі розв'язки запам'ятовуватися. Це можна продовжувати до виконання критерію зупинки. Вибір іншої початкової множини I_0 можна здійснити шляхом вибору однієї з «рівноцінних» множин околу $O(I)$ або іншим чином.

Формування початкової множини I_0, знаходження множин $I(G(List_{j_q}))$ та перегляд околу $O(I)$ в наближеному алгоритмі можна організувати наступним чином.

Початкова множина I_0. Початкову множину можна генерувати рандомізовано, але забезпечуючи при цьому, щоб вона була незалежною. Якщо з деякою вершиною із множини I_0 зв'язані інші вершини, то останні однозначно не можуть бути занесені до множини I_0. Якщо до початкової множини занести всі можливі вершини, то вона буде максимально незалежною по включенню.

Знаходження множин $I(G(List_{j_q}))$ можна виконати шляхом послідовного рандомізованого вибору до множини $I(G(List_{j_q}))$ та умовного видалення разом

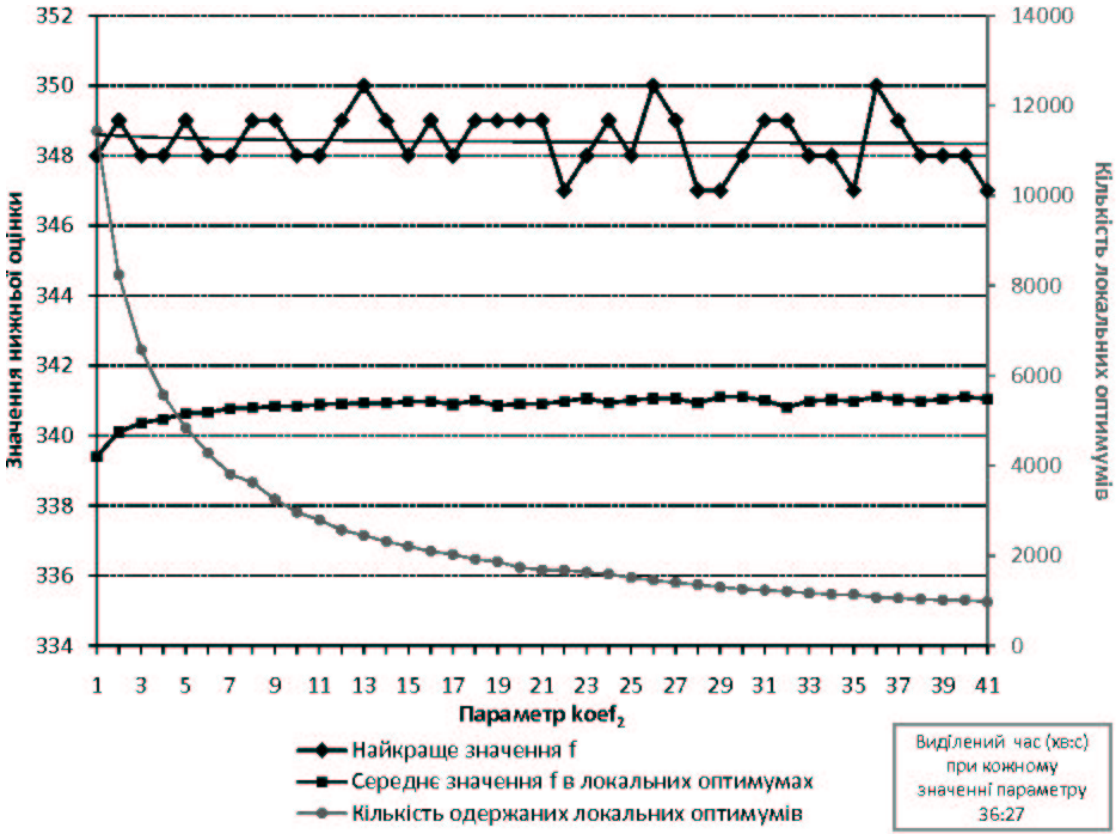

Рис. 3. Залежність від параметра $koef_2$ результатів роботи наближеного алгоритму (V) при $n = 12$

Таблиця 4.
Нижні оцінки кодів, одержані застосуванням наближеного алгоритму при модифікації (V), $koef_2 = 3,5$

n	Назва графу, що відповідає	Виділений час (хв:с:мс)	Нижня оцінка		c	d	e	f	g
			a	b					
4	1zc16	00:00:333	**4**	**4**	1 826	00:00:000	4,00	1 826	1
5	1zc32	00:01:000	**6**	**6**	2 211	00:00:000	6,00	2 211	3
6	1zc64	00:03:000	**12**	**12**	4 798	00:00:000	12,00	4 798	11
7	1zc128	00:09:000	**18**	**18**	2 960	00:00:000	18,00	2 960	15
8	1zc256	00:27:000	**36**	**36**	5 167	00:00:000	35,54	6 399	60
9	1zc512	01:21:000	**62**	**62**	3 365	00:00:016	61,42	4 339	83
10	1zc1024	04:03:000	**112**	**112**	6	00:24:812	108,30	4 900	252
11	1zc2048	12:09:000	**198**	195	1	08:28:688	187,73	4 889	585
12	1zc4096	36:27:000	**379**	348	3	03:43:407	340,47	3 496	855

a — відома, b — одержана, c — к-ть повторень найкращого розв'язку, d — затрачений час на одержання першого найкращого розв'язку (хв:с:мс), e — середнє значення цільової функції в локальних оптимумах, f — к-ть одержаних локальних оптимумів, g — к-ть ітерацій, 10^5

з околом вершин найменшого степеня з графу $G(List_{j_q})$. Такий спосіб вибору виявився ефективним для графів невеликих розмірностей, якими є графи $G(List_{j_q})$.

Перегляд околу $O(I)$ відбувається рандомізовано. Локальний оптимум I^L знайдено, якщо в окружності $O(I^L)$ немає жодної максимальної по включенню множини з к-тю вершин більшою ніж в I^L.

Якщо локальний оптимум знайдено і критерій зупинки не виконується, то наближений алгоритм можна запустити знов. Критерієм для зупинки може слугувати обмеження по часу, по кількості ітерацій, достатня кількість вершин незалежної множини, або інша умова, можлива їх комбінація.

Даний алгоритм було детально вивчено та відтестовано. Крім того, здійснено модифікації даного алгоритму, які було досліджено при різних значеннях параметрів. Слід відмітити, що робота алгоритму суттєво покращувалася при виборі вершин найменшого степеня.

У зв'язку з громіздкістю описання модифікацій алгоритму, результатів програмної реалізації та дослідження самого алгоритму та його модифікацій, наведемо тільки графічне зображення (рис. 3) залежності результатів роботи алгоритму при модифікації (V) від одного з параметрів ($koef_2$) при $n = 12$ та узагальнену табл. 4 результатів по модифікації (V). В результаті застосування наближеного алгоритму (по загальній схемі) та його модифікацій досягнуто відомі нижні оцінки при $n \leqslant 10$. Ефективність алгоритму підтверджує той факт, що за допомогою нього було досягнуто $[1, 3]$ відомі нижні оцінки і для $n \leqslant 11$, для $n = 12, 13$ одержано нові нижні оцінки.

В табл. 5 для порівняння зібрано найкращі значення цільової функції, отримані при застосуванні до поставленої задачі розглянутих вище алгоритмів: метод вектора спаду, вибір вершин найменшого або малого степеня, наближений алгоритм та його модифікації. З цієї таблиці ми маємо наглядну інформацію

Таблиця 5.

Нижні оцінки кодів, одержані розглянутими вище алгоритмами

n	Назва графу, що відповідає	Виділений час (хв:с:мс)	Одержана нижня оцінка								Відома нижня оцінка
			Метод вектора спаду	a		Наближений алгоритм (I) та його модифікації (II-IV)					
				(I)	(II)	(I)	(II)	(III)	(IV)	(V)	
4	1zc16	00:00:333	**4**	**4**	**4**	**4**	**4**	**4**	**4**	**4**	**4**
5	1zc32	00:01:000	**6**	**6**	**6**	**6**	**6**	**6**	**6**	**6**	**6**
6	1zc64	00:03:000	**12**	**12**	**12**	**12**	**12**	**12**	**12**	**12**	**12**
7	1zc128	00:09:000	**18**	16	**18**	**18**	**18**	**18**	**18**	**18**	**18**
8	1zc256	00:27:000	33	**36**	**36**	**36**	**36**	**36**	**36**	**36**	**36**
9	1zc512	01:21:000	55	**62**	**62**	**62**	**62**	**62**	**62**	**62**	**62**
10	1zc1024	04:03:000	92	108	108	107	108	109	**112**	**112**	**112**
11	1zc2048	12:09:000	158	188	186	173	189	178	194	196	**198**
12	1zc4096	36:27:000	278	340	334	308	342	302	348	350	**379**

а — вибір вершин найменшого (I) або малого (II) степеня

щодо алгоритмів. Наближений алгоритм (особливо при модифікації V) дав найкращі результати серед розглянутих вище.

В роботі також розроблено наближений алгоритм, в якому було враховано переваги вище розглянутих алгоритмів. Даний алгоритм було застосовано до поставленої задачі при $n = \overline{4, 13}$ і він дав найкращі результати в роботі. На відміну від вище розглянутих, за допомогою останнього в роботі було досягнуто відомі нижні оцінки і при $n = \overline{11, 12}$, а при $n = 13$ покращено відому нижню оцінку до значення 704.

В табл. 6 представлено нижні оцінки об'єму перешкодозахищених кодів при $n = \overline{10, 13}$, що одержані даним алгоритмом. Було виконано 10 спроб одержання найкращого значення нижньої оцінки.

Таблиця 6.

Нижні оцінки, одержані ще одним наближеним алгоритмом

n	Назва графу, що відповідає	Відома нижня оцінка	Найкраща одержана нижня оцінка ($\|I\|^*$)	$\|I\|_{\text{сер}}$	$t_{\text{сер}}(\|I\|_{\text{сер}})$, с	$DISP(t(\|I\|_{\text{сер}}))$
10	lzc1024	**112**	**112**	112	193,908	33307,050
11	lzc2048	**198**	**198**	198	507,191	102705,666
12	lzc4096	**379**	**379**	379	1880,859	3598555,791
13	lzc8192	**701**	**704**	703,9	9977,937	50394295,224

Заключення

Отже, в роботі одержано відомі [1–4, 7–9] нижні оцінки об'єму перешкодозахищених кодів, що коригують одну помилку в Z-каналі при $n \leqslant 12$. Причому, при $n \leqslant 10$ ці оцінки є точними [2,8–11], тобто, вже доведено, що такі об' єми відповідних перешкодозахищених кодів є максимальними. Крім того, покращено відому нижню оцінку об'єму таких кодів при $n = 13$ від значення 701 [3,8] до 704.

1. *Шило В. П.* Новые нижние оценки объема помехозащищенных кодов для Z-канала // Кибернетика и систем. анализ. - 2002. - № 1. - С. 19-23.
2. *Butenko S., Pardalos P. M., Sergienko I., Shylo V., and Stetsyuk P.* Estimating the size of correcting codes using extremal graph problems // In Optimization: Structure and Applications (Edited by C. Pearce and Emma Hunt) Springer (2009).
3. *Сергиенко И. В., Шило В. П.* Задачи дискретной оптимизации: проблемы, методы решения, исследования. - Киев: Наук. думка, 2003. -264 с.
4. *Сергиенко И. В., Шило В. П., Стецюк П. И.* Приближенный алгоритм решения задачи нахождения максимального независимого множества // Компьютерная математика. - Киев: Ин-т кибернетики им. В.М. Глушкова НАН Украины, 2000. - С. 4-20.
5. *Сергиенко И. В.* Математические модели и методы решения задач дискретной оптимизации. - Киев: Наук. думка, 1988. - 471 с.
6. *Abello J., Butenko S., Pardalos P., and Resende M.* Finding Independent Sets in a Graph Using Continuous Multivariable Polynomial Formulations // Journal of Global Optimization, Vol. 21, pp. 111-137, 2001. (http://www.mgvis.com/Papers/Comb_Alg_Comp/jogo2001a.pdf, ост. відв. 27.10.2009)
7. *Andrade D. V., Resende M. G. C., and Werneck R. F.* Fast local search for the maximum independent set problem // Proceedings of 7th International Workshop on Experimental Algorithms (WEA 2008), LNCS. Cape Cod, MA, 2008. (http://www.optimization-online.org/DB_FILE/2008/02/1898.pdf, ост. відв. 27.10.2009)
8. *Sergienko I. V. and Shylo V. P.* Problems of discrete optimization: challenges and main approaches to solve them // Cybernetics and Systems Analysis, Vol. 42, No. 4, (2006). (http://www.springerlink.com/content/e02h0653n0166144/fulltext.pdf, ост. відв. 01.11.2009)
9. *Sloane N. J. A.* Challenge problems: Independent sets in graphs, 2000. http://www.research.att.com/ñjas/doc/graphs.html (ост. відв. 01.11.2009).
10. *Шило В. П.* Точный алгоритм решения задачи нахождения максимального объема помехозащищенного кода. // Компьютерная математика. - Киев: Ин-т кибернетики им. В.М. Глушкова НАН Украины, 2002. - Вып. 2. - С. 50-57.
11. *Шило В. П.* Точное решение задачи построения помехозащищённого кода максимального объёма // Компьютерная математика. - Киев: Ин-т кибернетики им. В.М. Глушкова НАН Украины, 2005. - № 2. - С. 145-152.

Одержано 04.11.2009