УДК 519.72

І.О. Завадський, к. ф.-м. н., доцент

Завадостійкі коди на основі скінченних автоматів як узагальнення згорткових кодів

Київський національний університет імені Тараса Шевченка, 03680, м. Київ, пр-т. Глушкова 4д, e-mail: zava@ukr.net

Igor O. Zavadskyi, Associate Professor, Ph. D.

The error-correcting codes by means of finite automatons as generalization of convolutional codes

Taras Shevchenko National University of Kyiv, 03680, Kyiv, Glushkova av., 4d

У роботі описано метод завадостійкого кодування на основі скінченних автоматів, що може розглядатися як узагальнення широко відомих згорткових кодів. Показано, що деякі коди на основі скінченних автоматів є ефективнішими за згорткові коди за тієї ж обчислювальної складності алгоритму декодування.

Ключові слова: завадостійке кодування, скінченний автомат, згортковий код.

The forward error correcting codes exploiting the finite automatons as core encoding and decoding unit are introduced. As is known, the convolutional codes can be represented in the form of finite automaton diagram as well, but such diagram has some restrictions, e.g. it should contain 2^n states and symbols written to output code should be calculated as the results of linear functions of state number and input symbol. It is shown in the paper how the efficiency of convolutional codes can be improved with no complexity cost if to avoid the linearity restriction. The computational experiment results are presented in the paper. It is shown that for some values of E_s/N_0 ratio the finite automaton code with semi-randomly generated markup using the algorithm presented can outperform the NASA convolutional code with the same number of states by 7–8%. Also some general principles of finite automaton codes efficiency estimation are outlined in the paper.

Key words: error correcting, finite automaton, convolutional codes.

Статтю представив д.ф.-м.н., проф. Анісімов А.В.

Одним із найпоширеніших різновидів завадостійких кодів є згорткові коди, що детально досліджені в [1]. Своїй популярності вони завдячують високій ефективності, простоті кодувального алгоритму та швидкості методів декодування, насамперед методу Вітербі. Однак, як буде показано в цій статті, згорткові коди є лише частинним випадком кодів на основі скінченних автоматів. Ми також сконструюємо коди на основі скінченних автоматів, які виправляють помилки ефективніше за найкращі зі згорткових кодів за тієї самої обчислювальної складності.

Кодування

Кодування, тобто перетворення вхідної послідовності бітів на ту, що передаватиметься каналом зв'язку, здійснюється скінченним автоматом, стани якого пронумеровані. Стану, із якого автомат починає роботу, надамо номер 0. На кожній ітерації автомат зчитує k бітів вхідного коду і, залежно від зчитаного значення, здійснює один із 2^k переходів у якийсь із інших станів, записуючи у вихідний код p бітів, де $p > k$. Отже, код, що генерується автоматом, має швидкість k / p.

На діаграмі автомату символи, які він зчитує для здійснення того чи іншого переходу, записуватимемо над стрілкою переходу перед скісною рискою, а символи вихідного коду, що генеруються автоматом, — після неї. З кожного стану автомата виходитиме 2^k стрілок переходів. Наприклад, на рис. 1 зображено такий автомат швидкості ½ з 4 станами. Якщо на вхід автомату надійде послідовність бітів 011010, то послідовність станів автомату під час її обробки буде такою: 0,0,1,3,3,2,1, а вихідний код таким: 00 11 01 10 01 00.

Декодування

Декодування прийнятої послідовності бітів можна здійснювати за методом, подібним до методу Вітербі для згорткових кодів [1].

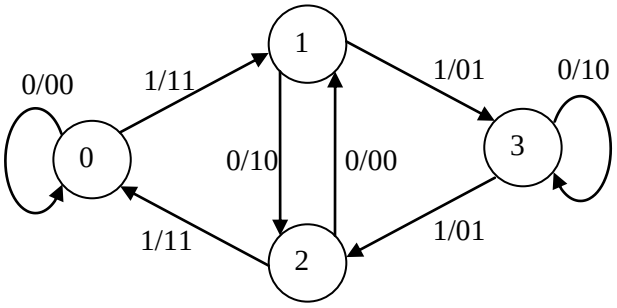


Рис. 1. Приклад кодувального автомата

Цей метод дає змогу знайти кодове слово (тобто слово, що може бути згенероване кодером), яке в певній описаній нижче метриці є найближчим до слова, отриманого кодувальним пристроєм. У разі жорсткого рішення (на вхід декодера надходять біти) такою метрикою є хемінгова відстань, однак ми розглядатимемо більш ефективне м'яке рішення: у каналі зв'язку на біти, що кодуються амплітудами 1 (одиничний біт) і –1 (нульовий біт), накладається адитивний білий гаусів шум, і на вході декодера отримуємо дійсні числа, що визначають вірогідність хибної інтерпретації того чи іншого сигналу як нульового чи одиничного біта.

Алгоритм декодування виконується у два проходи: прямий та зворотний. На *i*-му кроці прямого проходу кожному зі станів *j*, у яких автомат може перебувати після *i*-го переходу під час кодування, приписується метрика

$$m_i^j = \max_{k \in K_j} \left(m_{i-1}^k + \delta\left(s_{kj}, r_i\right) \right) \qquad (1)$$

Тут K_j — множина станів, з яких є переходи в стан *j*, s_{kj} — рядок із *p* символів, які записує автомат у вихідний код на переході з *k*-го стану в *j*-й, r_i — набір із *p* дійсних чисел — сигналів на вході декодера, що відповідають *i*-му переходу, а $\delta\left(s_{kj}, r_i\right) = \sum_{t=1}^{p} \delta_t\left(s_{kj}^t, r_i^t\right)$, де

$$\delta_t\left(s_{kj}^t, r_i^t\right) = \begin{cases} -1 - r_i^t, \text{якщо } s_{kj}^t = '0'; \\ r_i^t - 1, \text{якщо } s_{kj}^t = '1'. \end{cases}$$

Під час декодування, як і під час кодування, автомат починає роботу в нульовому стані, тому покладемо $m_0^0 = 0$.

Зображення всіх можливих переходів автомата на кількох ітераціях називатимемо *трелісом*, за аналогією зі згортковими кодами. Треліс автомату з рис. 1 для 5 перших ітерацій зображено на рис. 2. Над стрілками зазначено символи, які автомат записує у вихідний код на відповідних переходах — зіставлення цих символів переходам називатимемо їх *маркуванням*. Біля станів вказано їхні метрики за припущення, що на вхід декодера надійшов вектор сигналів із такими амплітудами: {–0,23; 0,34; 0,79; 0,05; 1,15; –1,03; –1,87; –0,71}. Жирним позначено стрілки, за якими обчислюються метрики, тобто переходи з таких станів *k*, для яких досягається максимум величини $m_{i-1}^k + \delta\left(s_{kj}, r_i\right)$.

На зворотному проході вибираємо стан з останньої ітерації з максимальною метрикою і будуємо на трелісі шлях, проходячи за жирними стрілками у зворотному напрямі, — називатимемо цей шлях *оптимальним*. Тобто в кожному стовпці треліса вибираємо по одному стану за таким принципом: якщо в *i*+1-му стовпці вибрано стан *j*, то в *i*-му стовпці вибираємо серед усіх станів, з яких є переходи у стан *j*, той, що максимізує метрику стану *j*. Якщо разом із формуванням оптимального шляху в кодувальному автоматі формувати також рядок із відповідних символів вхідного коду, дописуючи їх щоразу до рядка зліва, то отримаємо результат декодування.

Якщо на певні сигнали накладено високий рівень шуму протилежної до самих сигналів величини (тобто на сигнал з амплітудою 1 великий від'ємний шум, а на сигнал з амплітудою –1 — великий додатний шум), то стан з максимальною метрикою на ітерації, коли обробляється відповідний сигнал, може бути визначено неправильно. Однак з високою ймовірністю автомат повернеться на правильний шлях протягом кількох наступних ітерацій і при цьому шлях відкоригується і в місці помилки також.

Так, припустимо, що зазначеним вище амплітудам {–0,23; 0,34; 0,79; 0,05; 1,15; –1,03; –1,87; –0,71} на вході декодера відповідає набір бітів 00 11 10 00 на виході кодувального пристрою, тобто кодувальний автомат протягом перших 4-х ітерацій пройшов через стани 0, 0, 1, 2, 1. Однак внаслідок похибок у каналі зв'язку амплітуда 2-го біта, яка початково дорівнювала –1, була сильно викривлена і через це на 2-й ітерації максимальну метрику має стан 2, а на 3-й — стан 3, хоча за відсутності похибок такими станами мали б бути 1 і 2 відповідно. Проте вже на 4-й ітерації ситуацію виправлено: максимальну метрику має стан 1 і, рухаючись назад за жирними стрілками, отримаємо правильну послідовність станів: 0, 0, 1, 2, 1.

Таким чином, щоб виправляти помилки в останніх бітах кодового слова, ще перед кодуванням бітової послідовності її потрібно доповнювати фіктивними бітами, наприклад нульовими. Достатньою довжиною цього доповнення є $5 \log_2 N$, де N — кількість станів

автомату. Це так зване «вікно», у якому працює автомат під час декодування, і його розмір дорівнює кількості стовпців треліса, які необхідно зберігати в пам'яті програми чи апаратного пристрою, які реалізують даний метод. Тобто використовується обсяг пам'яті, що пропорційний величині $N \log_2 N$.

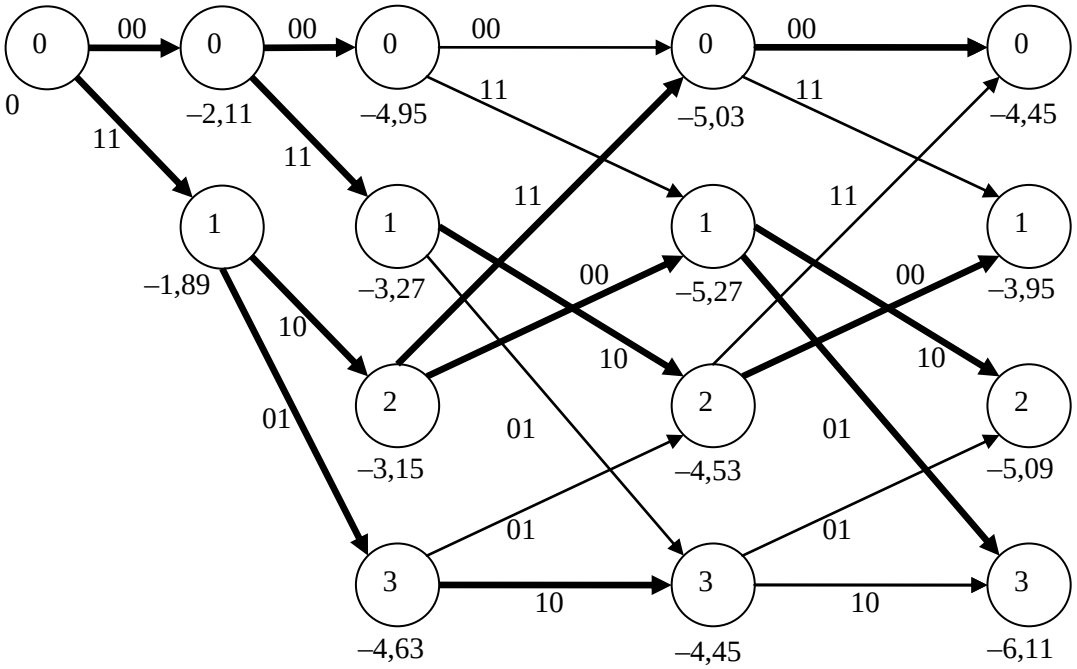
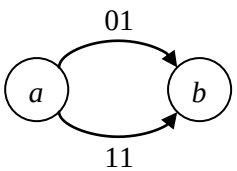


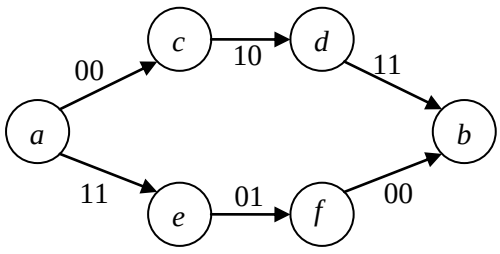
Рис. 2. Треліс декодувального скінченного автомата

Конструювання ефективних кодувальних автоматів

Як зазначалося в попередньому розділі, небажаною ситуацією під час декодування є та, коли внаслідок похибок у каналі зв'язку на певній ітерації змінюється стан, який має максимальну метрику. Однак із високою ймовірністю через певну кількість ітерацій максимальну метрику знову матиме належний стан і, якщо жирні стрілки в трелісі не змінилися, оптимальний шлях на зворотному проході буде побудовано правильно. Гірше, коли жирні стрілки змінюються — тоді навіть в разі повернення максимальної метрики до належного стану на зворотному проході певну ділянку шляху може бути побудовано неправильно і декодоване слово міститиме помилки. Ймовірність такої ситуації буде тим вищою, чим меншою є хемінгова відстань між словами, що генеруються під час проходу на трелісі або в автоматі різними шляхами, які починаються в тому самому стані a та закінчуються в тому самому стані b (називатимемо їх *напівциклами*). Наприклад, якщо з певного стану два переходи ведуть в однаковий стан і хемінгова відстань між відповідними словами у вихідному коді становить 1 (рис. 3а), то значна похибка лише в одному біті може призвести до того, що між цими станами буде встановлено хибний оптимальний шлях і на наступних ітераціях цю ситуацію жодним чином не буде виправлено. Натомість, якщо хемінгова відстань між шляхами напівциклу дорівнює 6 (рис. 3б), то велику похибку в одному сигналі, найімовірніше, буде виправлено більш коректними амплітудами інших 5 сигналів і оптимальний шлях до стану b залишиться коректним.



а



б

Рис. 3. Структури переходів між станами скінченного автомату, що можуть призводити до помилок під час декодування: помилка високоймовріна (а); ймовірність помилки значно нижча (б)

Надалі для спрощення розглядатимемо лише автомати швидкості ½, які на кожному переході зчитують 1 біт вхідного коду й записують 2 біти у вихідний код. Однак всі викладені результати неважко узагальнити й на випадок довільних кодувальних автоматів.

Отже, структури, подібні зображеним на рис. 3а, є неприпустимими. Більш точно, неприпустимою є наявність у графі автомату напівциклів, хемінгова відстань між шляхами яких є невеликою. Далі ми опишемо різні способи усунення цих небажаних структур, але насамперед вкажемо 2 простих принципи, дотримуватися яких під час створення автомату дуже просто, і які гарантуватимуть хемінгову відстань не менше 4 для напівциклів довжини не менше 2:

- переходи з того самого стану мають маркуватися рядками, хемінгова відстань між якими становить 2;
- переходи у той самий стан мають маркуватися рядками, хемінгова відстань між якими становить 2.

Оскільки автомат швидкості ½ на кожному переході зчитує 1 біт, то з будь-якого його стану має бути 2 переходи, що відповідають значенням цього біта. Перша умова означає, що рядки символів, які генеруються на цих переходах, мають бути протилежними: 00 та 11 або 01 та 10, а друга — що в кожен стан входить не більше 2 стрілок з протилежним маркуванням: 00 та 11 або 01 та 10. Зауважимо, що якщо в певний стан є менше 2 переходів, то в якийсь інший стан має бути більше 2 переходів, а отже, для дотримання другої умови необхідно, щоб у кожний стан було точно 2 переходи. Таким чином, можна сформулювати певну «структурну» властивість ефективного кодувального автомату швидкості ½: з будь-якого його стану та в будь-який його стан є точно 2 переходи.

Спробуємо застосувати для конструювання ефективного кодувального автомату ймовірнісний підхід: візьмемо автомат, переходи якого згенеровано випадковим чином, але з дотриманням умови, що з будь-якого його стану та в будь-який його стан є точно 2 переходи, і опишемо алгоритм, що дає змогу позбутися в цьому графі напівциклів довжини, меншої за деяке число k (довжиною шляху вважатимемо кількість переходів на ньому).

1. $t \leftarrow 1$.
2. Якщо $t=k$, кінець алгоритму. Інакше переглядаємо всі стани автомату. Якщо з певного стану a автомата є два шляхи довжини t, які завершуються в тому самому стані b, виконуємо кроки 3–5.
3. Нехай якийсь зі згаданих у п. 2 шляхів закінчується переходом $c \to b$. Видаляємо цей перехід.
4. Вибираємо довільний стан автомата d. Припустимо, що до цього стану ведуть переходи $e \to d$ і $f \to d$. Видаляємо будь-який із цих переходів, наприклад $e \to d$.
5. Створюємо переходи $c \to d$ і $e \to b$.
6. Якщо на кроці 2 було виявлено хоча б один стан, шляхи з якого вимагають виправлення на кроках 3–5, повертаємось на крок 2, інакше збільшуємо значення t на 1 і повертаємось на крок 2.

Якщо значення k буде завеликим, алгоритм може працювати надто довго чи взагалі не завершити роботу, оскільки під час усунення напівциклу на кроках 3–5 з великою ймовірністю утворюватиметься один чи кілька нових напівциклів. Для заданого значення k експериментальним шляхом можна визначити найменшу кількість станів автомата, для якої час роботи алгоритму усунення напівциклів довжини k буде задовільним. Відповідні співвідношення наведено в табл. 1.

Табл. 1. Співвідношення між максимальною довжиною напівциклів, які можна усунути, та найменшою кількістю станів автомата

Максимальна довжина напівциклів, які можна усунути	Найменша кількість станів автомата
2	5–7
3	15–18
4	164–167
5	1085–1090

Автомати, створені за наведеним вище ймовірнісним алгоритмом, за належного маркування, принцип якого буде наведено нижче, генерують коди з достатньо високою завадостійкістю. Однак цей показник можна покращити, якщо замість автомата, граф якого створено переважно випадково, використовувати автомат спеціальної структури, яку

опишемо далі. Цей автомат, що генерує код швидкості ½, матиме 2^m станів; у кожен стан та з кожного стану буде по 2 переходи. Якщо не враховувати переходи зі станів із номерами від 2^{m-1}до 2^m–1, то граф має структуру бінарного дерева, у кожну вершину якого входить одна дуга. Друга дуга до кожної вершини спрямована від одного з листків цього дерева (рис. 4).

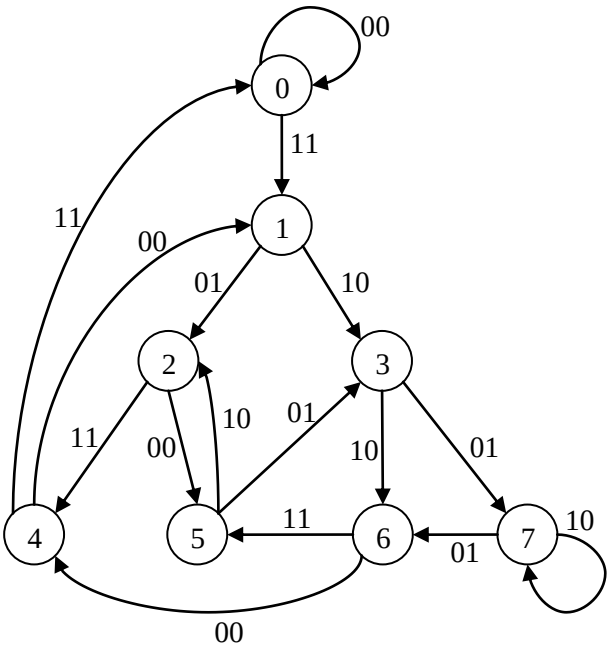


Рис. 4. Деревоподібний кодувальний автомат

Опишемо алгоритм створення структури деревоподібного автомата.

1. Для кожного стану *a* перевіряємо, чи є з нього шляхи.
2. Створюємо переходи 0→0 та 0→1.
3. Створюємо деревоподібну структуру переходів зі станів із номерами від 1 до 2^{m-1}–1: з кожного такого стану *i* створюємо переходи в стани 2*i* та 2*i*+1.
4. Для кожного стану *i*, $2^{m-1} \le i < 2^m$, створюємо переходи в стани $2i–2^m$ та $2i–2^m+1$.

Автомат, побудований за цим алгоритмом, має важливу властивість, що встановлюється лемою 1.

Лема 1. Деревоподібний автомат з 2^m вершинами, побудований за наведеним вище алгоритмом, не містить напівциклів довжини, що менша або дорівнює *m*.

Доведення. Розглянемо подання номера *i* деякого стану автомата у двійковому вигляді. З цього стану може бути виконано перехід до станів 2*i*, 2*i*+1, $2i–2^m$ або $2i–2^m+1$. Перехід до стану 2*i* означає зсув двійкового подання номера *i* вліво на 1 біт, перехід до стану 2*i*+1 — зсув вліво на 1 біт та інверсію наймолодшого (першого) біта, перехід до стану $2i–2^m$ — зсув вліво на 1 біт та інверсію найстаршого (*m*+1-го) біта, а перехід до стану $2i–2^m+1$ — зсув вліво на 1 біт та інверсію найстаршого й наймолодшого бітів. Отже, який би перехід не здійснювався, той біт, який був у двійковому поданні номера стану *i* наймолодшим, наступної ітерації зсунеться вліво на 1 позицію, але його значення не зміниться. Більше того, його значення не зміниться протягом *m* ітерацій, аж поки він не стане найстаршим (*m*+1-м). Тепер зауважимо, що наймолодші біти номерів станів, у які може бути виконано перехід із будь-якого стану *i*, завжди відрізняються. А отже, відрізнятимуться другі біти номерів станів, у які можна потрапити різними шляхами зі стану *i* за два переходи, треті біти номерів станів, у які можна потрапити різними шляхами зі стану *i* за три переходи і т.д. Таким чином, з певного стану неможливо потрапити в один той самий інший стан різними шляхами довжини менше, ніж *m*+1.

Лему доведено.

Результат, що встановлюється лемою 1 для деревоподібного кодувального автомату, значно кращий за аналогічний показник для випадкових автоматів, наведений у табл. 1. Це підтверджується і експериментальними оцінками завадостійкості кодів, що генеруються цими автоматами.

Що стосується маркування скінченного автомата, нижче буде наведено алгоритм, який дає змогу дотримуватися трьох умов.

1. Переходи з того самого стану маркуються протилежними рядками, тобто 00 та 11 або 01 та 10.
2. Переходи у той самий стан маркуються протилежними рядками.
3. Якщо мають місце переходи *a*→*b* і *a*→*c*, то переходи зі станів *b* і *c* повинні маркуватися різнотипними маркуваннями, тобто якщо переходи з одного з цих станів маркуються як 00 і 11, то переходи з іншого стану мають маркуватися як 01 і 10.

Виконання перших двох умов, як зазначалося вище, гарантує, що хемінгова відстань між шляхами будь-яких напівциклів, що містять не менше 2 ребер, становитиме не менше 4, а третя умова, як не важко побачити,

гарантує, що хемінгова відстань між шляхами будь-яких напівциклів довжини не менше 3 становить не менше 5. Отже, опишемо алгоритм маркування автомату довільної структури, який гарантує дотримання цих трьох принципів.

1. Переходи зі стану 0 маркуємо як 00 та 11, а переходи зі стану 1 — як 01 та 10.

2. Переглядаємо всі стани. Якщо точно один із переходів у стан маркований, маркуємо інший перехід у цей стан протилежним рядком.

3. Переглядаємо всі стани. Якщо точно один із переходів зі стану маркований, маркуємо інший перехід із цього стану протилежним рядком.

4. Повторюємо кроки 3 і 4 доти, доки з'являтимуться нові маркування.

5. Переглядаємо всі стани. Якщо переходи із деякого стану *a* не марковані, маркуємо їх протилежними рядками так, щоб не порушувався принцип 3 ані для переходів у стан *a*, ані для переходів з цього стану, і повертаємося на крок 3. Якщо жодного немаркованого стану не знайдено, маркування завершено.

Зауважимо, що неважливо, яким саме значенням вхідного біта зіставляти одну чи другу стрілку, що виходять зі стану автомата: оскільки значення бітів у вхідному потоці рівноймовірні, згаданий вибір не впливає на завадостійкість коду.

Зв'язок зі згортковими кодами

Розглянемо несистематичний згортковий код швидкості ½ з пам'яттю *m*. Кожної ітерації його кодувальний пристрій зчитує з вхідного потоку *m* двійкових символів, які позначимо $x_t,\ldots, x_{t+m-1}$ та обчислює на їх основі дві лінійні функції:

$$y_1 = x_{i_1} \oplus x_{i_2} \oplus \ldots \oplus x_{i_k}$$

$$y_2 = x_{j_1} \oplus x_{j_2} \oplus \ldots \oplus x_{j_l}$$

Значення цих функцій, тобто біти y_1 та y_2, записуються у вихідний код. Наступної ітерації кодувальний пристрій зчитує символи $x_{t+1},\ldots, x_{t+m+1}$, знову обчислює на них ті самі лінійні функції, записує 2 біти в код на виході і т.д. Якщо інтерпретувати біти $x_t,\ldots, x_{t+m-1}$ як двійкове подання номера *i* стану скінченного автомата, то цей автомат матиме 2^m вершин, а наступне двійкове число x_{t+1} ,…, x_{t+m} дорівнюватиме:

$2i$, якщо $x_t = 0$ і $x_{t+m} = 0$;

$2i + 1$, якщо $x_t = 0$ і $x_{t+m} = 1$;

$2i - 2^{m+1}$, якщо $x_t = 1$ і $x_{t+m} = 0$;

$2i - 2^{m+1} + 1$, якщо $x_t = 1$ і $x_{t+m} = 1$.

Це означає, що діаграма станів згорткового коду швидкості ½ має таку саму деревоподібну структуру, як описано в попередньому розділі. Однак згортковий код має додаткове обмеження стосовно маркування: біти маркерів переходів є значеннями двох лінійних функцій, що залежать від номера стану переходу. У довільному коді з деревоподібним автоматом, який марковано за наведеним у попередньому розділі алгоритмом, ці функції не обов'язково є лінійними і в деяких випадках можуть забезпечувати вищу завадостійкість, ніж найкращі згорткові коди відповідної швидкості та пам'яті. Так, у табл. 2 порівнюється завадостійкість найвідомішого й найефективнішого згорткового коду швидкості ½ та пам'яті 6 — коду NASA (171,133) із найкращим зі 100 деревоподібних кодів, автомати яких мають ту саму кількість вершин і були марковані за наведеним вище ймовірнісним алгоритмом. Як видно, завадостійкість запропонованого нами коду для деяких значень E_b/N_0 дещо вища. Чисельний експеримент проводився для м'якого рішення декодера на основі алгоритму Вітербі на неквантованому каналі BPSK.

Табл. 2. Порівняння завадостійкості кодів

E_b/N_0, Дб	BER, NASA (171,133)	BER, Finite automaton code
–1	0,302	0,293
0	0,1551	0,1432
1	0,1309	0,1309
2	0,005	0,0049

Слід особливо підкреслити, що виграш в ефективності виправлення помилок досягається без жодних додаткових витрат часу або пам'яті декодера порівняно з декодуванням згорткового коду за методом Вітербі.

Список літератури

1. *Johannesson R., Zigangirov K. S.* Fundamentals of Convolutional Coding, IEEE Press, 1999. – 400 p.

References

1. Johannesson R., Zigangirov K. S. (1999), *Fundamentals of Convolutional Coding,* IEEE Press. – 400 p.

Надійшла до редколегії 27.06.2014