ПРОГРАММНАЯ РЕАЛИЗАЦИЯ ДЕКОДЕРА СУДАНА ДЛЯ КОДОВ РИДА-СОЛОМОНА

© 2007 г. Н.В. Ремизов

В настоящее время для повышения надежности передачи информации широко используются помехоустойчивые коды Рида-Соломона. Дальнейшее развитие этого направления привело к разработке списочного декодирования, основанного на работах Судана [1].

Целью данной работы является исследование возможности программной реализации декодера Судана для кодов Рида-Соломона и численный анализ полученных результатов.

Пусть *GF(q)* – поле Галуа, $(\alpha_0,...,\alpha_{n-1})$ – фиксированный набор различных элементов из *GF(q), q* – мощность поля, $n \le q - 1$. Будем называть его опорным множеством. Процесс кодирования заключается в следующем: на вход подается к-мерный вектор $f = (f_0, \dots, f_{k-1}), f_i \quad GF(q)$, далее происходит вычисление полинома $p(x) = f_0 + f_1 x + \dots + f_{k-1}x_{k-1}$ в точках α_i, $i = 0 \dots n-1$. На выходе получаем n-мерный вектор $u=(p(\alpha_0), \dots, p(\alpha_{n-1}))$.

Линейный Код Рида-Соломона является МДР-кодом, то есть достигает границы Синглтона $d = n - k + 1$, где d – минимальное кодовое расстояние. Поэтому код способен исправлять $t = \left\lfloor \frac{n-k}{2} \right\rfloor$ ошибок.

Пусть C – кодирующее отображение, C: $F_q^k \to F_q^n$. Разобьем пространство F_q^n на сферы с центрами в кодовых словах радиуса t. Эти сферы, очевидно, не пересекаются. Обычный декодер анализирует пришедшее по каналу слово y и находит тот центр сферы радиуса t, в которую попало данное слово. И выдает соответствующее кодовое слово.

1. Списочный декодер Гурусвами-Судана. Теоретические основы списочного GS-декодера

Будем следовать описанию списочного декодера, приведенного в [1].

Рассмотрим последовательные шаги алгоритма. Пусть F_q – поле Галуа, C: $F_q^k \to F_q^n$ - кодирующее отображение, $(\alpha_0, \dots, \alpha_{n-1})$ - опорное множество кода Рида-Соломона, $y = (y_0, y_1, \dots, y_{n-1})$ – пришедшее по каналу слово.

Шаг 1. Строится полином от двух переменных Q(x, y) степени D по переменной x и степени L по переменной y, удовлетворяющий следующим условиям:

- $Q(\alpha i, yi) = 0$, $i = 0 \ldots n - 1$.
- $Q(x, y) \not\equiv 0$.

Шаг 2. Находятся все делители полинома *Q(x, y)* вида

$y - p(x)$, $\deg p(x) < k$, причем $p(\alpha_i) = y_i$ по крайней мере для $n - t$ значений $i = 0 \ldots n - 1$.

Параметры L и D в [1] определяются следующим образом: $L = \sqrt{\frac{n}{k}}$, $D = \sqrt{nk}$.

1.1. Интерполяция

В работе [1] М. Судан предлагает такой способ построения многочлена. Представим полином $Q(x, y) = \sum_{i,j} a_{ij} x^i y^j$ в виде:

$$Q(x, y) = \sum_k q_k(x) y^k =$$
$$= (q_{00} + q_{01}x + \ldots + q_{0D}x^D) + (q_{10} + q_{11}x + \ldots + q_{1D}x^D)y + \ldots + (q_{00} + q_{01}x + \ldots + q_{0D}x^D)y^L$$

Подставляя вместо переменной x элементы опорного множества, а вместо переменной y – компоненты пришедшего по каналу слова, получаем однородную систему линейных алгебраических уравнений относительно q_{ij}.

$$\begin{cases} q_{00}z_{00}^0 + q_{01}z_{01}^0 \ldots + q_{0D}z_{0D}^0 + q_{10}z_{10}^0 + q_{11}z_{11}^0 + \ldots \\ \qquad + q_{1D}z_{1D}^0 + \ldots + q_{L0}z_{L0}^0 + q_{L1}z_{L1}^0 + \ldots + q_{LD}z_{LD}^0 = 0 \\ q_{00}z_{00}^{n-1} + q_{01}z_{01}^{n-1} \ldots + q_{0D}z_{0D}^{n-1} + q_{10}z_{10}^{n-1} + q_{11}z_{11}^{n-1} + \ldots \\ \qquad + q_{1D}z_{1D}^{n-1} + \ldots + q_{L0}z_{L0}^{n-1} + q_{L1}z_{L1}^{n-1} + \ldots + q_{LD}z_{LD}^{n-1} = 0, \end{cases}$$

где $z_{ij}^k = \alpha_k^i y_k^j$, $0 \le k \le n - 1$, $0 \le i \le D$, $0 \le j \le L$.

Ненулевое решение существует, т.к. количество неизвестных, больше, чем количество уравнений. (Степень многочлена $Q(x, y)$ по переменной x должна не больше, чем $\frac{n}{L}$, и по переменной y – не больше, чем L, тогда количество неиз-

вестных $(L+1)(\left\lfloor \frac{n}{L} \right\rfloor +1) > n$). Это один из вариантов. В работе [2] МакЭлиас приводит альтернативный алгоритм решения проблемы интерполяции – алгоритм Кюттера. Приоритетом при выборе играет время – если метод Гаусса требует $O(n^3)$ (порядка n^3) операций, то алгоритм Кюттера – $O(n^2)$ (порядка n^2) операций.

1.2. Факторизация. Алгоритм Рота-Рукенштейна

Пусть F – поле, может быть, конечное, $F[x, y]$ – кольцо многочленов от 2-х переменных над полем F. Каждый элемент $Q(x, y)$ $F[x, y]$ можно представить в виде суммы

$$Q(x, y) = \sum_{i,j} a_{ij} x^i y^j ,$$

где не все a_{ij} равны 0.

Определение. Пусть $w = (u, v)$, $u \geq 0$, $v \geq 0$, $u + v \neq 0$. Взвешенной степенью монома $x^i y^j$ называется число $\deg_w x^i y^j = iu + jv$.

Обозначим через $XY = \{x^i y^j, i = 0, 1 \ldots n \ldots, j = 0, 1 \ldots n \ldots\}$. Используя понятие взвешенной степени, введем на множестве XY отношения порядка “ $<_{\text{wlex}}$” и “ $<_{\text{wrevlex}}$” .

Определение. Будем говорить, что мономы упорядочены согласно отношению *w-lex* (*w-lexicographic order*), если

$x^{i_1} y^{j_1} < x^{i_2} y^{j_2}$,

если $ui_1 + vj_1 < ui_2 + vj_2$, или $ui_1 + vj_1 = ui_2 + vj_2$ и $i_1 < i_2$.

Определение. Будем говорить, что мономы упорядочены согласно отношению *w-revlex* (*w-reverse lexicographic order*), т.е.

$x^{i_1} y^{j_1} < x^{i_2} y^{j_2}$,

если $ui_1 + vj_1 < ui_2 + vj_2$, или $ui_1 + vj_1 = ui_2 + vj_2$ и $i_1 > i_2$.

Пусть $w = (u, v)$, “ < ” – некоторое отношение порядка (“$<_{\text{wlex}}$” или “$<_{\text{wrevlex}}$”) на множестве XY. Следовательно, можно упорядочить элементы множества:

$$Const = \phi_0(x, y) < \phi_1(x, y) < \phi_2(x, y) < \ldots,$$

где $\phi_k(x, y) = x_i y_j$ XY.

Тогда каждый полином $Q(x, y)$ можно записать в уникальной форме:

$$Q(x, y) = \sum_{j=0}^{J} a_j \phi_j(x, y)$$

Число J называется рангом полинома $Q(x, y)$, Rank $(Q) = J$. Моном $\phi_J(x, y)$ называется ведущим мономом и обозначается $LM(Q)$.

Замечание. Рассмотрим $F[x, y]$ – кольцо многочленов от двух переменных надо полем F. P, Q $F[x, y]$. Если $LM(P)=LM(Q)$, то говорят, что $P \equiv Q$.

Определение. Взвешенной степенью полинома $Q(x, y)$ называется число $\deg_w Q = \max_i\{\deg_w \phi_i(x, y) \mid a_i \neq 0\}$.

Свойства взвешенной степени: $w = (u, v)$,

1. $\deg_w 0 = -\infty$;
2. $\deg_w(PQ) = \deg_w P + \deg_w Q$;
3. $\deg_w (P + Q) \leq \max(\deg_w P , \deg_w Q)$.

Напомним, на втором шаге GS-декодера необходимо найти все делители полинома $Q(x, y)$ $F[x, y]$ вида $y - p(x)$, $p(x)$ $F[x]$, $\deg p(x) \leq k$-1.

Определение. Будем говорить, что полином $Q(x, y)$ имеет ноль в точке (a, b), если

$$Q(a, b) = 0.$$

Определение. Будем говорить, что полином $Q(x, y) = \sum_{i,j} a_{ij} x^i y^j$ имеет ноль кратности, или порядка m в точке (0, 0), и писать

$$\mathrm{Ord}(Q: 0, 0) = m,$$

если в записи полинома все $a_{ij} = 0$, $i + j < m$.

Теорема. Предположим, $p(x)$ $F_{k-1}[x]$, $Q(x, y)$ $F[x , y]$, $w = (1, k\text{-}1)_{\mathrm{lex}}$, $n = \sum_a \mathrm{ord}(Q : a, p(a))$. Если $n > \deg_w Q$, то $(y - p(x)) \mid Q(x, y)$.

Лемма 1. Пусть $w = (1, k\text{-}1)$. Если $p(x)$ $F_{k-1}[x]$, тогда

$$\deg Q(x, y) < \deg_w Q(x, y).$$

Лемма 2. $Q(x, p(x)) = 0$ в том и только том случае, когда $(y - p(x)) \mid Q(x, y)$.

Для удобства формулировок назовем $p(x)$ y-корнем полинома $Q(x, y)$.

Алгоритм Рота-Рукенштейна (РР-алгоритм) получает на вход полином от двух переменных $Q(x,y)$ и выдает список y-корней этого полинома. Рассмотрим данный алгоритм подробнее.

Найдем такую наибольшую степень m переменной x так, что $x^m \mid Q(x, y)$, но $x^{m+1} \nmid Q(x, y)$, и сократим. Получим полином

$$\text{«Q(x, y)»} = \frac{Q(x, y)}{x^m}.$$

Таким образом, если исходный полином $Q(x, y)$ в точке $(0,y)$ мог обращаться в 0, то $\text{«}Q(0, y)\text{»} \neq 0$.

Покажем, процесс последовательного определения коэффициентов a_i полинома $p(x)$.

$$p(x) = a_0 + a_1x + \ldots + a_{k-1}x_{k-1}.$$

Лемма. Если $(y - p(x)) \mid Q(x, y)$, тогда $y = p(0) = a_0$ – решение уравнения:

$$Q_0(x, y)=0,$$

где $Q_0(x, y) = \text{«}Q(x, y)\text{»}$.

Построим последовательность полиномов:

$P_0(x) = p(x)$, $Q_0(x, y) = \text{«}Q(x, y)\text{»}$. Для $j \geq 0$ получаем:

$$p_j(x) = (p_{j-1}(x) - p_{j-1}(0))/x = a_j + a_{j+1}x + \ldots + a_{k-j}x_{k-j} \ ,$$
$$T_j(x, y) = Q_{j-1}(x, xy + a_{j-1}), \qquad (1)$$
$$Q_j(x, y) = \text{«}T_j(x, y)\text{»}.$$

Теорема. Пусть дан полином $p(x) = a_0 + a_1x + \ldots + a_{k-1}x_{k-1}$ $F_{k-1}[x]$, $Q(x, y)$ $F[x, y]$. Определим последовательности полиномов по формулам (1). Тогда $\forall j \geq 1$ $(y - p(x)) \mid Q(x, y)$ в том и только том случае, когда $(y - p_j(x)) \mid Q_j(x, y)$.

Следствие. Если $y \mid Q_k(x, y)$, т. е. если $Q_k(x, 0) = 0$, тогда

$p(x) = a_0 + a_1x + \ldots + a_{k-1}x_{k-1}$ – y-корень полинома $Q(x, y)$.

Алгоритм строит дерево коэффициентов. Каждый ярус дерева “символизирует” степень переменной x((-1)-ярус введен для удобства). В вершинах i-го яруса находятся коэффициенты при степенях x^i возможных y-корней полинома $Q(x, y)$. С помощью алгоритма “поиска в глубину” производится обход дерева.

2. Моделирование и численный анализ GS-декодера

Рассмотрим примеры работы декодера Судана. Пусть:

$GF(q)$ – поле Галуа,

k – размерность кода;

n – длина кода;

t – количество гарантировано исправляемых ошибок;

x – информационный полином;

y – кодовое слово;

List – список y-корней.

Так как мы будем работать над полями характеристики 2 (char($GF(q)$)=2), то для наглядности переводим двоичный формат в десятичный, т. е. все коэффициенты в приведенных ниже примерах следует рассматривать как двоичные числа в десятичной записи. (Пр.: $f = (1, 3)_{10}$, что соответствует f=$(001, 011)_2$.)

1. $GF(2^3)$, k = 2, n = 5, t = 1;
   f = (1, 3) ;
   x = (1, 2, 7, 4, 6) ;
   список y-корней
   List: 1 3.

   _____

Вносим аддитивную ошибку:

1а. e = (0, 0, 2, 0, 0), тогда y = x + e, y = (1, 2, 5, 4, 6),
   List: 1 3.

1б. e = (0, 0, 2, 0, 1), тогда y = x + e, y = (1, 2, 5, 4, 7) ;
   List: пуст.

   _____

В 1а декодер выдал правильное кодовое слово, в 1б – ошибок больше, чем может исправить декодер.

2. $GF(2^3)$, k = 3, n = 5, t = 1;
   f = (1, 3 ,5);
   x = (1, 7, 5, 3, 5);
   List: 1 3 5.

Вносим аддитивную ошибку

2а. e: = (0, 0, 0, 0, 5) ,тогда y = x +e, y = (1, 7, 5, 3, 5);
   f = (7, 7, 7);
   x = (7, 7, 3, 3, 2);
   List: 7 7 7.

   _____

2б. e = (0, 0, 6, 0, 2), тогда y = x + e, y = (7, 7, 3, 5, 3, 0);
   List: 7 6 6.

Из чего можно сделать вывод, что декодер попал в сферу с центром в другом кодовом слове. Что вполне логично, так как количество исправляемых ошибок равно 1, а в данном примере их было внесено 2.

3. $GF(2^5)$, k =8, n = 32, t =12;
f = (29, 0, 2, 4, 6, 8, 10, 12);
x = (29, 19, 23, 11, 9, 9, 5, 10, 3, 10, 1, 29, 23, 9, 11, 29, 4, 2, 22, 22, 5, 0, 18, 12, 22, 17, 14, 8, 1, 28, 4, 5);
List: 29 0 2 4 6 8 10 12.

_____

Вносим аддитивную ошибку:

3а. e = (20, 26, 30, 2, 0, 0, 12, 3, 10, 3, 8, 20, 0, 0, 0, 0 ,0 ,0 ,0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0), w(e) = 12, тогда y = x + e,
y = (9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 23, 9, 11, 29, 4, 2, 22, 22, 5, 0, 18, 12, 22, 17, 14, 8, 1, 28, 4, 5);
List: 9 0 0 0 0 0 0 0 0;
29 0 2 4 6 8 10 12.

_____

3б. e = (21, 27, 31, 3, 1, 1, 13, 2, 11, 2, 9, 21, 31, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0), w(e) = 13; тогда y = x + e,
y = (8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 9, 11, 29, 4, 2, 22, 22, 5, 0, 18, 12, 22, 17, 14, 8, 1, 28, 4, 5);
List: 8 0 0 0 0 0 0 0 0;
29 0 2 4 6 8 10 12.

_____

3в. e = (29, 0, 21, 0, 0, 0, 3, 0, 11, 0, 11, 0, 27, 0, 5, 0, 20, 0, 0, 0, 17, 0, 4, 0, 14, 0, 20, 0, 29, 0, 26, 0), w(e) = 14; тогда y = x + e;
y = (14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 11, 29, 4, 2, 22, 22, 5, 0, 18, 12, 22, 17, 14, 8, 1, 28, 4, 5);
List: 29 0 2 4 6 8 10 12.

_____

3г. e = (29, 0, 21, 0, 13, 0, 3, 0, 11, 0, 11, 0, 27, 0, 5, 0, 20, 0, 0, 0, 17, 0, 4, 0, 14, 0, 20, 0, 29, 0, 26, 0), w(e) = 15; тогда y = x + e;

y = (14, 19, 21, 11, 0, 9, 19, 10, 26, 10, 17, 29, 24, 9, 15, 29, 22, 2, 22, 22, 20, 0, 27, 12, 18, 17, 25, 8, 16, 28, 23, 5);
List: пустой

_____

Список пуст, декодер не выдал ни одного слова. Следовательно, можно заключить, что максимальное количество исправляемых GS-декодером ошибок для кода Рида-Соломона размерности 8 и длины 32 равно 14.

Заключение

Списочный декодер позволяет исправлять большее количество ошибок, составляя список, в котором обязательно присутствует истинное информационное слово. В случае большего числа ошибок, декодер выдает либо ошибочное кодовое слово, либо пустой список.

В результате проведенной работы был программно реализован списочный декодер Гурусвами-Судана, проведена численная проверка работы декодера. Как известно, обычный вероятностный декодер для кодов Рида-Соломона позволяет исправлять $t = \left\lfloor \frac{n-k}{2} \right\rfloor$ ошибок. Списочный GS-декодер исправляет больше ошибок, в зависимости от размерности и длины кода.

Литература

1. *Madhu Sudan*, Lectures “Algorithmic Introduction to Coding Theory”, 2001.
2. *R.J. McEliece*, The Guruswami-Sudan Decoding Algorithm for Reed-Solomon codes, May 15, 2003.
3. *Лидл Р., Нидеррайер Г.* Конечные поля, в 2-х т. / Пер. с англ. М.: Мир, 1988. – 430 с.
4. *Берлекэмп Э.* Алгебраическая теория кодирования / Пер. с англ. М.: Мир, 1971.