Современные проблемы дистанционного зондирования Земли из космоса. 2014. Т. 11. № 2. С. 138-151

Методы ускорения алгоритмов декодирования символьных кодов

В.В. Золотарёв[1], И.В. Чулков[1], Г.В. Овечкин[1], Д.Ж. Сатыбалдина[2]

[1]Институт космических исследований РАН, Москва 117997, Россия
E-mail: zolotasd@yandex.ru
[2]Евразийский национальный университет им. Л.Н. Гумилева
Астана 010008, Республика Казахстан
E-mail: satybaldina_dzh@enu.kz

В некоторых системах передачи и хранения данных предпочтительнее использовать недвоичные помехоустойчивые коды. Показано, что известные символьные помехоустойчивые коды, такие как коды Рида-Соломона и q-ичные низкоплотностные коды, обладают низкой эффективностью или высокой сложностью реализации. Также рассмотрены недвоичные многопороговые декодеры (qМПД) символьных самоортогональных кодов (qСОК). Показано, что qМПД выполняет почти оптимальное декодирование qСОК с низким уровнем размножения ошибок всего лишь с линейной зависимостью сложности реализации от длины кода. Анализируется сложность реализации символьного порогового элемента (qПЭ), являющегося единственным вычислительно сложным блоком qМПД. Показано, что сложность обычной программной реализации qПЭ пропорциональна d^2, где d – кодовое расстояние. Предложены несколько методов ускорения работы qПЭ, обладающие разными требованиями к оперативной памяти. Данные методы обеспечивают существенное уменьшение числа выполняемых арифметических операций по сравнению с неоптимизированной реализацией qПЭ. Лучший из предложенных методов обеспечивает линейную зависимость числа выполняемых операций от кодового расстояния применяемого кода. В результате оказывается возможным увеличить быстродействие qМПД для типичных параметров кодов в 2 и более раз по сравнению со стандартным алгоритмом работы qПЭ без потери в эффективности декодирования.

Ключевые слова: итеративное декодирование, недвоичные (символьные) многопороговые декодеры, недвоичные самоортогональные коды, пороговый элемент, сложность декодирования.

Введение

Помехоустойчивое кодирование используется для исправления ошибок, возникающих при передаче данных по каналам с шумами. Основное внимание в литературе уделяется двоичным помехоустойчивым кодам, работающим с данными на уровне отдельных битов. Однако во многих цифровых системах часто удобнее работать с данными, имеющими байтовую структуру. Например, удобнее работать с байтами в системах хранения больших объемов информации (оптические диски и др. носители). В подобных системах для защиты данных от ошибок целесообразно применение недвоичных помехоустойчивых кодов. В настоящее время наиболее широкое применение среди недвоичных кодов нашли коды Рида-Соломона (РС), для которых существуют алгебраические алгоритмы декодирования (Berlakamp, 1968), позволяющие исправлять до половины кодового расстояния ошибок, а также более сложные алгоритмы (Wu, 2008), обеспечивающие исправление несколько большего числа ошибок. Однако данные методы из-за высокой сложности реализации позволяют декодировать только короткие и поэтому малоэффективные РС коды. В

последнее время многие специалисты занимаются развитием декодеров недвоичных низкоплотностных кодов (*q*LDPC), которые способны обеспечить очень высокую эффективность (Zhang, Pfister, 2007). Однако сложность таких декодеров, особенно при большом размере символа, все еще остается слишком высокой для практического применения.

Особое место среди недвоичных алгоритмов коррекции ошибок занимают рассматриваемые далее символьные (недвоичные) самоортогональные коды и соответствующие им специальные высокоскоростные символьные многопороговые декодеры (*q*МПД) (Золотарев, Овечкин, 2010), являющиеся развитием двоичных многопороговых декодеров (МПД) (Золотарев и др., 2008). Двоичные МПД обеспечивают близкое к оптимальному декодирование со всего лишь линейной зависимостью сложности реализации от длины используемых кодов (Золотарев и др., 2006). Большой интерес к МПД проявляется не только в России (Ullah et al., 2010). Представленные в (Золотарев, Овечкин, 2010) результаты исследований показывают, что *q*МПД существенно перекрывают по своей эффективности РС коды и практически реализуемые *q*LDPC коды, оставаясь столь же простыми в реализации, как и их прототипы – двоичные МПД**.** Большую роль играет и отсутствие необходимости использования умножений в недвоичных полях при кодировании и декодировании, а также полная независимость длин символьных кодов от размеров используемых символов. Поэтому такие коды обязательно найдут широкое применение в сфере обработки, хранения и передачи больших объемов аудио- , видео- и других типов данных.

Оставшаяся часть статьи организована следующим образом. В секции 1 изложены базовые сведения о *q*МПД. В секции 2 обсуждаются вопросы ускорения *q*МПД, приведены различные алгоритмы работы недвоичного порогового элемента и выполнен анализ их вычислительной сложности. В заключении приведены основные выводы.

1. Недвоичные многопороговые декодеры

Опишем принципы работы *q*МПД при декодировании недвоичных самоортогональных кодов (*q*СОК). Описание дано для *q*-ичного симметричного канала (*q*СК) с размером алфавита q, $q > 2$, и вероятностью искажения символов p_0.

Пусть задан линейный недвоичный систематический сверточный или блоковый самоортогональный код, проверочная матрица **H** которого имеет такой же вид, как и в двоичном случае (Massey, 1963), т.е. состоит только из нулей и единиц, за исключением того, что вместо 1 в единичной подматрице будут –1, т.е. **H** = [**P** : –**I**]. Здесь **P –** подматрица, определяемая порождающим полиномом для *q*СОК; **I** – единичная подматрица.

Порождающая матрица такого кода будет иметь вид $\mathbf{G} = [\mathbf{I} : \mathbf{P}^{\mathrm{T}}]$. Данный код может использоваться при любом размере алфавита q.

Отметим, что для заданного таким образом qСОК при выполнении кодирования и декодирования требуются только операции сложения и вычитания по модулю q. Вычисления в недвоичных полях в данном случае не используются.

Пример схемы, реализующей операцию кодирования блоковым qСОК, заданного порождающим полиномом $g(x) = 1+x+x^4+x^6$, представлен на *рис. 1*. Такой код характеризуется параметрами: длина кода n=26 символов, длина информационной части кода k=13 символов, кодовая скорость R=1/2, кодовое расстояние d=5.

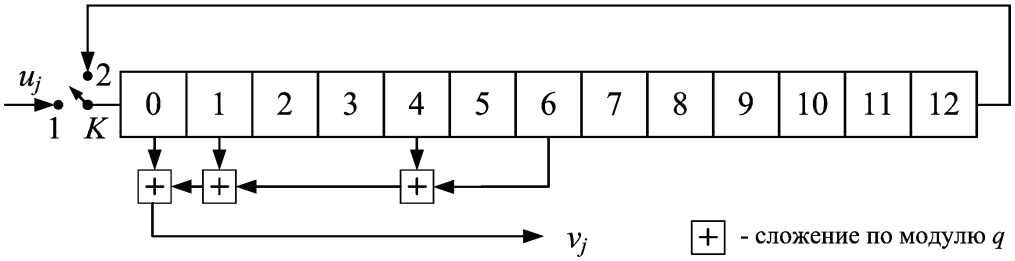

Рис. 1. Кодер для блокового qСОК, заданного полиномом $g(x) = 1+x+x^4+x^6$

Пусть кодер выполнил кодирование информационного вектора $\boldsymbol{U}$ и получил кодовый вектор $\boldsymbol{A} = [\boldsymbol{U}, \boldsymbol{V}]$, где $\boldsymbol{V} = \boldsymbol{U} \cdot \mathbf{G}$. Отметим, что здесь и далее при выполнении операций умножения, сложения и вычитания векторов и матриц используется модульная арифметика. После передачи кодового вектора $\boldsymbol{A}$ длины n с k информационными символами по qСК в декодер поступает вектор $\boldsymbol{Q}$, отличающийся, вообще говоря, от исходного кодового вектора из-за искажений в канале: $\boldsymbol{Q} = \boldsymbol{A} + \boldsymbol{E}$, где $\boldsymbol{E}$ – вектор шума канала типа qСК.

Алгоритм работы qМПД при декодировании вектора $\boldsymbol{Q}$ заключается в следующем (Золотарев и др., 2012).

1. Вычисляется вектор синдрома $\boldsymbol{S} = \mathbf{H} \cdot \boldsymbol{Q}^{\mathrm{T}}$. Обнуляется разностный регистр $\boldsymbol{D}$. В данном регистре будут отмечаться измененные декодером информационные символы. Отметим, что число ненулевых элементов векторов $\boldsymbol{D}$ и $\boldsymbol{S}$ всегда будет определять расстояние между принятым из канала сообщением $\boldsymbol{Q}$ и кодовым словом, являющимся текущим решением qМПД. И задачей декодера является найти такое кодовое слово, для которого число ненулевых элементов векторов $\boldsymbol{D}$ и $\boldsymbol{S}$ будет минимальным. Данный шаг полностью соответствует двоичному случаю.

2. Для произвольно взятого q-ичного декодируемого информационного символа i_j принятого сообщения подсчитывается число двух наиболее часто встречающихся значений проверок s_j вектора синдрома $\boldsymbol{S}$ из общего числа всех проверок, относящихся к сим-

волу i_j, а также символа d_j вектора $\boldsymbol{D}$, соответствующего символу i_j. Пусть значения этих двух проверок равны h_0 и h_1, а их количество равно m_0 и m_1 соответственно, причем $m_0 \geq m_1$. Данный шаг является аналогом процедуры получения суммы на пороговом элементе в двоичном МПД.

3. Если $m_0 - m_1 > T$, где T – значение порога (некоторое целое неотрицательное число), то из i_j, d_j и всех проверок относительно i_j вычитается оценка ошибки, равная h_0. Данный шаг является аналогом сравнения суммы с порогом в двоичном декодере и изменения декодируемого символа и коррекции через обратную связь всех символов синдрома, являющихся проверками для декодируемого символа.

4. Происходит выбор нового i_m, $m \neq j$, и осуществляется переход к п. 2.

Такие попытки декодирования по пп. 2…4 могут быть повторены для каждого символа принятого сообщения несколько раз (Золотарев и др., 2012). Заметим, что при реализации алгоритма qМПД, как и в двоичном случае, удобно все информационные символы перебирать последовательно, а останавливать процедуру декодирования после фиксированного числа попыток (итераций) коррекции ошибок или если при очередной такой итерации ни один из символов не изменил своего значения. Пример схемной реализации qМПД для кодера с *рис. 1* представлен на *рис. 2*.

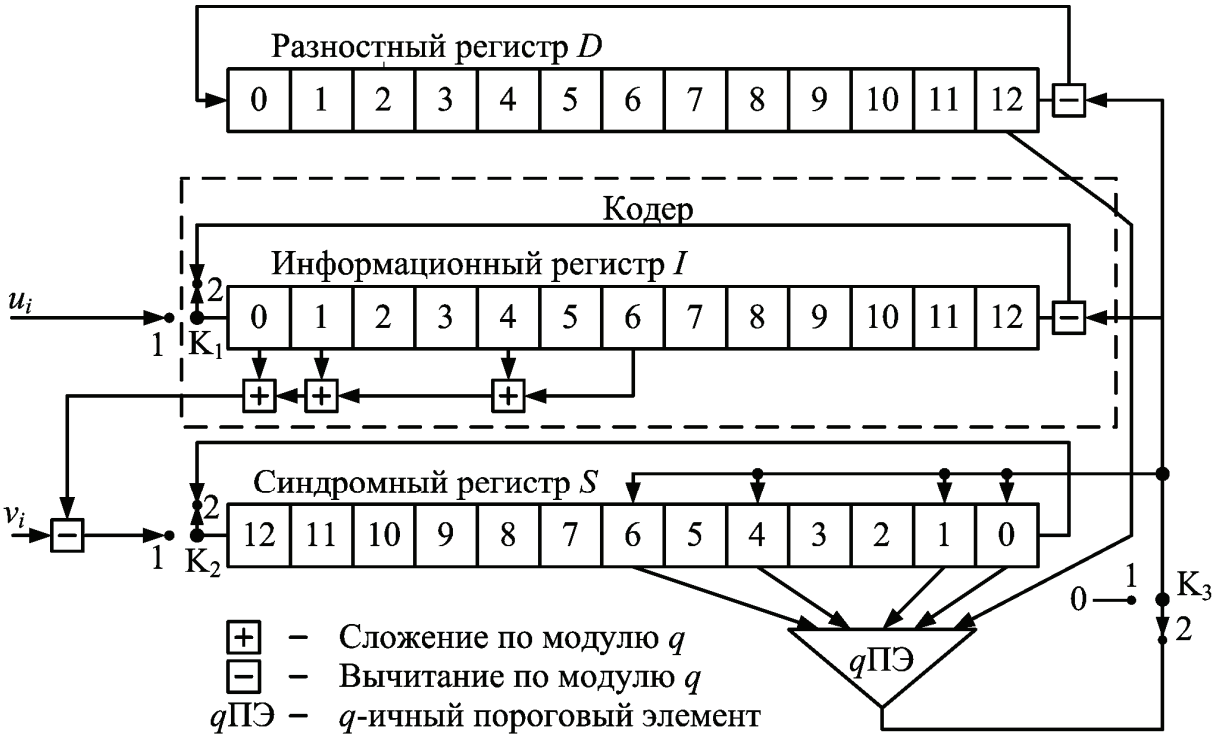

Рис. 2. qМПД для блокового qСОК

Для описанного алгоритма qМПД справедлива следующая теорема.

Теорема. Пусть декодер реализует алгоритм qМПД для описанного выше кода. Тогда при каждом изменении декодируемых символов происходит переход к более правдоподобному решению по сравнению с предыдущими решениями декодера.

Доказательство теоремы дано в (Золотарев и др., 2012). При доказательстве показывается, что суммарный вес Хемминга синдромного и разностного регистров при изменении декодируемого символа в соответствии с вышеописанным алгоритмом qМПД строго уменьшается.

Отметим два наиболее существенных момента, характеризующих предложенный алгоритм. Во-первых, как и в случае двоичных кодов, нельзя утверждать, что улучшение решения qМПД при многократных попытках декодирования будет иметь место до тех пор, пока не будет достигнуто решение оптимального декодера. На самом деле и в блоковых, и в сверточных кодах возможны конфигурации ошибок, не исправляемые в qМПД, но которые могут быть исправлены в оптимальном декодере. Поэтому основной способ повышения эффективности qМПД состоит в поиске кодов, в которых такие неисправляемые конфигурации ошибок довольно редки даже при большом уровне шума. Вопросы выбора таких кодов подробно рассмотрены в (Золотарев и др., 2012).

Другим важнейшим моментом является то, что по сравнению с традиционным подходом к мажоритарным схемам, в qМПД для изменения декодируемого символа достаточно наличие не абсолютного, а только относительно строгого большинства проверок, как это следует из условия $m_0 – m_1 > T$. Например, в qСОК с $d = 9$ ошибка в декодируемом символе будет исправлена даже в том случае, если из девяти его проверок (включая и символ d_j разностного регистра) правильными будут только две, а остальные семь – ошибочными! Этого невозможно представить для двоичных кодов, а для qМПД такая ситуация типична.

Эти свойства существенно расширяют возможности недвоичного многопорогового алгоритма при работе в больших шумах, сохраняющего, как следует из представленного описания, всего лишь линейную зависимость сложности реализации от длины кода.

Следует отметить, что при правильном выборе кодов qМПД оказывается способен обеспечить их близкое к оптимальному декодирование. В результате его характеристики помехоустойчивости оказываются много лучше, чем характеристики даже многократно более сложных декодеров для кодов Рида-Соломона и практически реализуемых недвоичных LDPC кодов. Например, в (Овечкин, Овечкин, 2009) показано, что с помощью достаточно простых каскадных кодов со скоростью 1/2, при декодировании которых используется qМПД, можно обеспечить вероятность ошибки декодирования порядка 10^{-9} при 27% байтовых ошибок в канале, что является в настоящее время недостижимым для других

методов коррекции ошибок в символьных данных. Высокую эффективность qМПД обеспечивает и при декодировании малоизбыточных самоортогональных кодов.

2. Методы ускорения работы порогового элемента в qМПД

При решении проблемы максимальной производительности qМПД, важную роль играет скорость работы его единственного активного узла – символьного порогового элемента (qПЭ). Рассмотрим возможные варианты его реальной работы при программной реализации, когда скорость обработки входного потока ошибок определяется числом выполняемых операций при декодировании каждого отдельного символа, поступившего из канала связи. Снова предположим, что анализируется линейный систематический блоковый или сверточный символьный самоортогональный код с размером алфавита q, кодовой скоростью $R < 1$ и минимальным кодовым расстоянием d. Выбор произвольно большого значения q предполагает, что в отдельных случаях, например, для той или иной формы программной реализации qПЭ в qМПД потребует очень большого объема требуемой памяти. Но во многих случаях qПЭ все же не потребует значительных объемов оперативной памяти процессора.

Ниже анализируется традиционная схема работы qПЭ, а затем рассмотрены другие варианты реализации программных версий qПЭ, некоторые из которых могут реально и существенно увеличить скорость его работы.

Стандартная реализация qПЭ

Рассмотрим для некоторого значения параметров q, d и R процесс принятия решения символьным пороговым элементом. Полагаем, что сначала рассматривается его работа при большом уровне шума. Это значит, что при типичном значении $d \geq 5$ и любых q, $q>>1$, и R, символы проверок для декодирования некоторого информационного символа i_j из вектора синдрома $\boldsymbol{S}$, сформированного в qМПД, имеют, в основном, различные значения. Практически для всех случаев большого шума, как легко проверить из правила работы qПЭ, для больших размеров символов алфавита совпадающие значения двух и более символов проверок наиболее часто бывают в тех случаях, когда результат декодирования оказывается правильным, т.е. именно эти проверки оказываются безошибочными, что и приводит к правильному решению qПЭ.

Оценим в такой постановке задачи число операций, которое необходимо выполнить qПЭ, работающим в соответствии с ранее описанным алгоритмом, чтобы определить

искомое единственное значение m_0 наиболее часто встречающихся проверок или вынести решение об отсутствии такого единственного числа.

Проанализируем стандартную схему работы qПЭ. Сначала необходимо обнулить d ячеек памяти во вспомогательных векторах $\boldsymbol{A}$ и $\boldsymbol{B}$ для дальнейших вычислений. В векторе $\boldsymbol{A}$ из d элементов будем хранить значения символов синдрома, а в векторе $\boldsymbol{B}$ такого же размера – число таких значений в просмотренных проверках. Далее начинаем последовательно выбирать проверки из множества из d проверок и помещать сведения о них в эти вектора. Для этого, заполняя ячейки векторов $\boldsymbol{A}$ и $\boldsymbol{B}$ с младших номеров, введем номер N текущей свободной ячейки массива. Каждая новая из d возможных проверок проверяется на то, что все предыдущие $\boldsymbol{N}$-1 ячеек содержат другие значения i_j. Как было обусловлено выше, обычно все значения проверок, в основном, различны. Значит, вектора $\boldsymbol{A}$ и $\boldsymbol{B}$ обычно будут чаще всего заполнены полностью. Вектор значений $\boldsymbol{A}$ будет заполнен полностью до размеров d значениями проверок, а вектор $\boldsymbol{B}$ в этом случае будет содержать только 1. В редких случаях наличия совпадающих значений проверок элементы вектора $\boldsymbol{B}$ будут иметь значения большие 1, но их будет при большом уровне шума совсем немного. Понятно, что в этих редких случаях заполнение векторов $\boldsymbol{A}$ и $\boldsymbol{B}$ будет несколько меньшим, чем до размера $\boldsymbol{d}$.

Из описанных процедур следует, что при ориентации на типичное максимальное заполнение рассматриваемых векторов общее число операций для этих вычислений составляет

$$N_1=4\cdot d + d\cdot(d-1)/2. \qquad (1)$$

Таким образом, заполнение рабочих векторов, в основном, имеет квадратичную сложность от кодового расстояния используемого кода. Для больших значений d эта зависимость будет определяющей.

Далее необходимо найти максимально часто встречающееся и при этом единственное значение проверок или решить, что его нет. Проверяются последовательно все ячейки в векторах $\boldsymbol{A}$ и $\boldsymbol{B}$. Каждая очередная ячейка сопоставляется с текущим самым часто встречающимся значением проверок и при равенстве или превышении текущего самого частого значения записывается как новое самое частое значение, а старое их значение переписывается во вторую рабочую ячейку. При этом для двух этих ячеек в их вспомогательные ячейки каждый раз записывается и частота появления этих значений в текущем наборе проверок. Общее число таких поисковых операций равно

$$N_2 = 5\cdot d. \qquad (2)$$

В случае, если иногда случается, что часть ячеек в векторе $\boldsymbol{B}$ содержат не 1, а немного большие числа, что при большом шуме бывает довольно редко, то перезапись значений

проверок и их числа происходит значительно реже. Поэтому оценка (2) вполне подходит в качестве верхней оценки для второго шага и мало отличается от реального числа операций.

Наконец, если оказалось, что в результате выполнения второй группы вычислений самое часто встречающееся значение проверок не равно 0 и разность между числом самых частых ошибок в первой вспомогательной ячейке и во второй превышает значение порога T для данного qПЭ, то все проверки и символы i_j и d_j надо изменить на m_0. Число требуемых для этого операций равно

$$N_3 = d+4. \qquad (3)$$

Таким образом, верхняя достаточно точная оценка числа операций для стандартной схемы qПЭ равна с учетом (1) – (3)

$$N_{\text{ст}} = 10\cdot d + d\cdot(d - 1) + 4. \qquad (4)$$

Как следует из (4), qПЭ выполняет 2 группы вычислений с линейной и квадратичной сложностью. Очевидно, что при большом значении d, например, d = 15…35, которое возможно при использовании длинных наиболее эффективных кодов, квадратичная компонента сложности будет определяющей.

Модель *q*ПЭ при неограниченной памяти

Реализация более быстрого qПЭ возможна в том случае, если будет найден алгоритм поиска единственного самого часто встречающегося значения проверок с линейной от параметра d сложностью. Рассмотрим один из таких возможных вариантов реализации qПЭ.

Пусть qПЭ применяется к коду с некоторыми параметрами q, R и d, как в предыдущем случае для стандартного варианта его реализации. Пусть первоначально будут обнулены все q ячеек некоторого рабочего вектора $\boldsymbol{A}$ и d ячеек во втором частотном векторе $\boldsymbol{B}$. Далее выбираются последовательно все d проверок декодируемого символа i_j и в ячейку вектора $\boldsymbol{A}$ с номером q_n, где q_n – значение очередной проверки, добавляется единичка. При этом такое же значение записывается в соответствующую ячейку вектора $\boldsymbol{B}$. Объем операций для этой группы вычислений равен

$$N_1 = q + 3\cdot d. \qquad (5)$$

Затем проверяется, не превышено ли в соответствующей ячейке вектора $\boldsymbol{A}$ текущее число наиболее часто встречаемых значений проверок. Эта группа вычислений уже полностью совпадает со второй группой вычислений для стандартной схемы и оценивается тем же выражением (2).

Очевидно, что процесс корректировки символов, если ее необходимость определил qПЭ, имеет ту же сложность, что и в первом случае, определяемую выражением (3).

Сделаем теперь весьма важное замечание. Если теперь перейти к декодированию нового информационного символа, то можно повторить уже проделанные для первого символа вычисления. Но в выражении (4) учтено, что для нового процесса коррекции очередного символа сначала надо обнулить вектор $\boldsymbol{A}$, который при больших значениях q может быть чрезмерно большим. Однако с другой стороны понятно, что этот вектор и так почти полностью обнулен, кроме тех d ячеек, в которые записывались значения проверок. Это значит, что на самом деле обнулять только именно те ячейки, в которые добавлялись значения проверок, которые одновременно записывались в соответствующие ячейки вектора $\boldsymbol{B}$. Значит, вектор $\boldsymbol{A}$ обнуляется полностью только один раз на старте процесса декодирования. А на каждом цикле надо обнулять в конце процесса декодирования каждого символа только d конкретных ячеек.

Тогда полная сложность qПЭ с неограниченной памятью становится равной

$$N_{\text{нп}}=4\cdot d + 5\cdot d + d + 4 = 10\cdot d + 4. \qquad (6)$$

Как видно из этого выражения, такая схема оказывается гораздо проще первой, так как имеет линейную от параметра d сложность.

Схема вычислений в qПЭ при ограниченной памяти

Предложенный выше qПЭ с линейной сложностью является очень важным результатом, который во многих реальных системах, использующих qМПД, позволяет в насколько раз увеличить быстродействие декодера только за счет более быстрой программной реализации qПЭ.

Однако при больших значениях q, соответствующих вполне реальным значениям, которые могут быть близкими к 10^9 при работе с 4-х байтовыми символами кодов, когда $q\sim2^{32}$, что вполне реально для многих компьютерных систем, выделение до миллиарда ячеек памяти для вектора $\boldsymbol{A}$ в qПЭ, конечно, недопустимо. Более того, большинство компьютеров легко оперирует восьмибайтовыми словами, что делает абсолютно необходимым отдельно решить вопросы работы qПЭ при сильных ограничениях на размеры памяти векторов типа $\boldsymbol{A}$ для qПЭ, работающих с многобайтовыми алфавитами. Ниже предложен один из возможных подходов именно такого типа. При этом мы будем стараться оставить сложность нового qПЭ на уровне линейной по отношению к числу входов qПЭ.

Пусть длина символа q используемого символьного кода составляет L байт. Преобразуем теперь бывший вектор $\boldsymbol{A}$ в новый уже двумерный массив $\mathbf{A}$ рассматриваемого qПЭ, состоящий из L подмассивов по 256 байтов. Конечно, эти подмассивы можно было сформировать, вообще говоря, не на байтовой структуре, т.е. не требовать, чтобы размеры массивов были бы равными именно 256 байтам. Однако используемый байтовый формат яв-

ляется наиболее употребительным. Пусть работа qПЭ нового типа с ограниченной памятью массива **A** будет начинаться с того, что в первый из L подмассивов символа d_j, относящегося к декодируемому символу i_j записывается в ячейку с номером, равным первому байту символа d_j, единичка, а в другие L подмассивов с этим же номером – остальные байты символа d_j. Этот же номер запишем первую ячейку массива **B** такого же размера, как и **A**. Затем будем брать последовательно остальные $d - 1$ проверок из синдромного регистра и проводить с ними следующие манипуляции. Если первый байт очередного символа проверки таков, что в первом подмассиве байт с номером, равным первому байту символа проверки равен 0, то во все L ячеек с этим номером записываются побайтово значение проверки, а в соответствующую ячейку в **B** – значение первого байта этой проверки.

В том случае, если для некоторой следующей проверки окажется, что в первом подмассиве массива **A** уже есть ранее записанная единичка или другое небольшое число, т.е. уже в двух или более символах проверок первые байты совпали, то выполняются следующие операции. Сравниваются символ проверки и весь текущий символ массива **A** (с номером первого байта проверки) длиной L байтов. Если эти символы длиной L байтов совпали, то соответствующая ячейка счетчика в **B** увеличивается на 1. А в случае, если они не совпали полностью, этот символ полностью записывается в дополнительном массиве **C**. Факт использования дополнительного массива **C** отмечается в массиве **D**. Для каждой новой проверки снова проверяется в первом подмассиве **A** первая она или нет.

Оценим теперь максимальное число операций, которое выполняет qПЭ для больших значений q. В наихудшем случае все d проверок имеют совпадающие первые байты в проверках и различные биты в остальных частях символа проверки. Тогда каждый раз проверяется байт проверки. А потом вся проверка сравнивается с другими уже записанными проверками. Общее число сравнений в этом случае будет $d\cdot(d - 1)/2$, что, конечно, опять весьма много. Однако вероятность появления таких проверок крайне мала и имеет порядок $256^{-(d-1)}$, что для больших значений d составляет величины много меньшие, чем даже 10^{-20}. Типичным же для больших значений q и d при большом уровне шума будет случай, когда есть 2 или три символа проверок, которые совпадут. Практически всегда это будет означать, что в совокупности этого набора из d проверок есть 2 или 3 правильных проверки, которые и будут самыми часто встречающимися. Их значения, в том числе и нулевые, и будут определять значения ошибок в декодируемых символах.

Таким образом, в последнем случае при больших значениях q и d общее число операций является переменным. Максимальное число вычислений по порядку величины близко к $d\cdot(d - 1)/2$, но достигается чрезвычайно редко. В реальности, когда совпадений

значений проверок нет или их не более 3-х, общее число операций не превышает $10 \cdot d$. Поскольку поиск максимально встречающегося числа и изменение символов при декодировании выполняется тут за то же число операций, что и при «бесконечной» памяти, общий средний объем операций третьего типа qПЭ составляет

$$N_{оп}=10 \cdot d + 5 \cdot d + d + 4 = 16 \cdot d + 4.$$

Отметим, что необходимая для работы такого qПЭ память составляет не более $M = (d + 3) \cdot 256$ ячеек размером L байт и, в некоторых случаях, может быть немного уменьшена за счет более рационального использования памяти. Очевидно, что в двух первых вариантах реализации qПЭ потребности в памяти еще меньше.

В случае, если qМПД проектируется для работы с двухбайтовыми числами, принципы работы всех трех вариантов qПЭ совершенно не меняются. Некоторое отличие появляется только у второго варианта декодера с неограниченной памятью. Чтобы реализовать его работу точно так же, как у байтового qПЭ, потребуется $2^{16} = 65536$ ячеек двухбайтовой памяти вместо 256. Вместе с тем можно указать, что по текущим возможностям элементной базы этот объем памяти чрезвычайно мал и ее выделение вполне реально. Это же относится и к третьему варианту qПЭ для больших q. Такой qПЭ можно реализовать на тех же принципах, что и байтовый qПЭ.

При переходе к 4-хбайтовым символам q (объем алфавита составляет порядка 10^9 знаков) целесообразно перейти к третьему типу порогов или использовать другие модификации qПЭ второго типа.

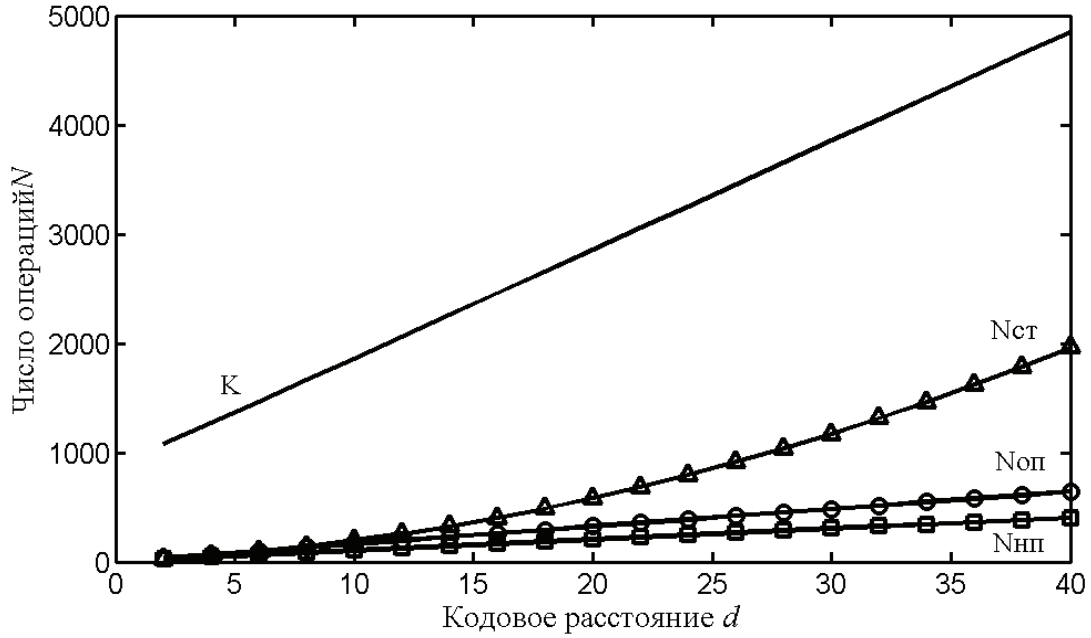

Рис. 3. Зависимость числа операций разных qПЭ от минимального кодового расстояния

График соотношения числа операций разных *q*ПЭ в зависимости от минимального кодового расстояния используемых кодов представлен на *рис. 3*. Как следует из его вида, с ростом значения *d* преимущество второй схемы с линейной сложностью (график «Nнп») над традиционной схемой (график «Nст») с квадратичной от *d* сложностью возрастает. График «K» показывает умноженное на 1000 отношение числа операций этих двух порогов, чтобы его можно было наблюдать на рисунке. Отметим, что для такого сложного устройства как *q*МПД даже увеличение скорости декодирования всего вдвое повышает и скорость всего процесса декодирования. Представленные результаты показывают, что при увеличении минимального кодового расстояния *d* до, например 40, реальное увеличение скорости *q*ПЭ достигает почти 5 раз. Поскольку *q*ПЭ – единственный активный элемент *q*МПД – простейшего, но очень эффективного по энергетике декодера, то общий рост производительности *q*МПД с «быстрым» *q*ПЭ будет близок при программной реализации к пятикратному.

Заключение

Приведенные результаты позволяют считать, что *q*МПД методы действительно относятся к уникальным алгоритмам, способным обеспечивать эффективное декодирование при большом уровне шума, выполняя очень небольшое число операций и достигая высочайших уровней достоверности передачи и хранения цифровой информации и скорости ее обработки в высокоскоростных линиях связи и в устройствах хранения больших объемов данных.

В статье получены важные результаты по модификации *q*ПЭ, позволившие перейти при большом уровне шума от квадратичной по *d* к линейной сложности реализации, что увеличило скорость работы *q*ПЭ до пяти раз при *d*~40. Использование *q*МПД для исправления ошибок в системах памяти с байтовой структурой данных может многократно повысить скорость работы программных версий алгоритмов кодирования и декодирования при реализации специальных версий декодеров с быстрыми пороговыми элементами.

Работа выполнена при финансовой поддержке Российского фонда фундаментальных исследований (грант №14-07-00859), Института космических исследований, гранта Президента Российской Федерации (грант МД-639.2014.9) и Комитета науки МОН Республики Казахстан. Большой объем дополнительной информации о многопороговых декодерах можно найти на веб-сайтах ИКИ РАН www.mtdbest.iki.rssi.ru и РГРТУ www.mtdbest.ru.

Литература

1. *Золотарев В.В., Зубарев Ю.Б., Овечкин Г.В.* Многопороговые декодеры и оптимизационная теория кодирования. М.: Горячая линия – Телеком, 2012. 239 с.
2. *Золотарев В.В., Зубарев Ю.Б., Овечкин Г.В.* Обзор методов помехоустойчивого кодирования с использованием многопороговых алгоритмов // Цифровая обработка сигналов. 2008. №1. С. 2–11.
3. *Золотарев В.В., Зубарев Ю.Б., Овечкин Г.В., Дмитриева Т.А.* Многопороговые алгоритмы для спутниковых сетей с оптимальными характеристиками // Электросвязь. 2006. № 10. С. 9–11.
4. *Золотарев В.В., Овечкин Г.В.* Эффективное многопороговое декодирование недвоичных кодов // Радиотехника и электроника. 2010. Т. 55. № 3. С. 324–329.
5. *Овечкин Г.В., Овечкин П.В.* Использование недвоичного многопорогового декодера в каскадных схемах коррекции ошибок // Вестник РГРТУ. 2009. № 4 (выпуск 30). С. 7–12.
6. *Berlekamp E. R.* Algebraic Coding Theory. McGraw-Hill. New York, 1968.
7. *Massey J.* Threshold decoding. M.I.T. Press. Cambridge. Massachusetts, 1963.
8. *Ullah M.A., Okada K., Ogivara H.* Multi-Stage Threshold Decoding for Self-Orthogonal Convolutional Codes. IEICE Trans. Fundamentals. 2010. Vol. E93-A. No.11. pp. 1932–1941.
9. *Wu C.* New list decoding algorithms for Reed-Solomon and BCH codes IEEE Transactions on Information Theory. 2008. Vol. 54. pp. 3611–3630.
10. *Zhang F., Pfister H.* List-Message Passing Achieves Capacity on the q-ary Symmetric Channel for Large q // Proc. IEEE Global Telecom. Conf. Washington, DC. Nov. 2007. pp. 283–287.

Methods for speeding up decoders for symbolic codes

V.V. Zolotarev[1], I.V. Chulkov[1], G.V. Ovechkin[1], D.J. Satibaldina[2]

[1]Space Research Institute RAS, Moscow 117997, Russia
E-mail: zolotasd@yandex.ru
[2]L.N. Gumilyov Eurasian National University
Astana 010008, Republic of Kazakhstan
E-mail: satybaldina_dzh@enu.kz

In some communication and data storage systems, non-binary (symbol) error-correcting codes are preferable use. The known symbol error-correction codes, such as Reed-Solomon codes and q-ary low-density parity-check codes, are shown to have either low symbol error rate performance or high implementation complexity. Symbolic multithreshold decoders (qMTD) for symbolic self orthogonal codes (qSOC) are also discussed. It is shown that qMTD performs almost optimal decoding for qSOC with low error-propagation at only linear complexity dependence on code length. Operations performed with symbolic threshold element (qTE) are analyzed. It is the most complex part of qMTD. The complexity of usual software implementation of qTE is proportional to d^2, where d is the code distance. Several methods for qTE speed improvement with different memory requirements are proposed. These methods provide essential decrease in the number of arithmetic operations in comparison with non-optimized implementation of qTE. The best of the proposed methods can provide the algorithmic complexity O(d). As a result, qMTD decoding speed is increased two or even more times for standard code parameters in comparison with usual qTE algorithm without performance loss.

Keywords: iterative decoding, symbolic (q-ary) multithreshold decoders, symbolic self-orthogonal codes, threshold element, decoding complexity.

References

1. Zolotarev V.V., Zubarev Yu.B., Ovechkin G.V. *Mnogoporogovye dekodery i optimizatsionnaya teoriya kodirovaniya* (Multithreshold decoders and optimizing coding theory), Moscow: Goryachaya liniya-Telekom, 2012, 239 p.
2. Zolotarev V.V., Zubarev Yu.B., Ovechkin G.V. Obzor metodov pomekhoustoichivogo kodirovaniya s ispol'zovaniem mnogoporogovykh algoritmov (Overview of error-correcting coding methods are used multithreshold algorithms), *Tsifrovaya obrabotka signalov*, 2008, No. 1, pp.2–11.

3. Zolotarev V.V., Zubarev Yu.B., Ovechkin G.V., Dmitrieva T.A. Mnogoporogovye algoritmy dlya sputnikovykh setei s optimal'nymi kharakteristikami (Multithreshold algorithms with optimal performance for satellite communications), *Elektrosvyaz'*, 2006, No. 10, pp. 9–11.
4. Zolotarev V.V., Ovechkin G.V. Effektivnoe mnogoporogovoe dekodirovanie nedvoichnykh kodov (Effective multithreshold decoding for non-binary codes), *Radiotekhnika i elektronika*, 2010, Vol. 55, No. 3, pp. 324–329.
5. Ovechkin G.V., Ovechkin P.V. Ispol'zovanie nedvoichnogo mnogoporogovogo dekodera v kaskadnykh skhemakh korrektsii oshibok (The using of non-binary multithreshold decoder in concatenated error-correction schemes), *Vestnik RGRTU*, 2009, No. 4, Issue 30, pp. 7–12.
6. Berlekamp E. R. *Algebraic Coding Theory*, McGraw-Hill, New York, 1968.
7. Massey J. *Threshold decoding*, M.I.T. Press, Cambridge, Massachusetts, 1963.
8. Ullah M.A., Okada K., Ogivara H. Multi-Stage Threshold Decoding for Self-Orthogonal Convolutional Codes, *IEICE Trans. Fundamentals*, Vol.E93-A, No. 11, pp. 1932–1941, Nov. 2010.
9. Wu C. New list decoding algorithms for Reed-Solomon and BCH codes*, IEEE Transactions on Information Theory*, Vol. 54, pp. 3611–3630, August 2008.
10. Zhang F., Pfister H. List-Message Passing Achieves Capacity on the q-ary Symmetric Channel for Large q, *Proc. IEEE Global Telecom. Conf.,* Washington, DC, pp. 283–287, Nov. 2007.